数据科学家
养成手册

高 扬◎编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

VISIT…

LANZAROTE Caliente.COM

内 容 简 介

作为认知科学的延伸，数据科学一方面应该越来越引起广大大数据工作者的重视，另一方面也要撩开自己的神秘面纱，以最为亲民的姿态和每位大数据工作者成为亲密无间的战友，为用科学的思维方式进行工作做好理论准备。本书从众多先贤及科学家的轶事讲起，以逐步归纳和递进的脉络总结出科学及数据科学应关注的要点，然后在生产的各个环节中对这些要点逐一进行讨论与落实，从更高、更广的视角回看科学及数据科学在各个生产环节的缩影。本书并不以高深的数学理论研究作为目的，也不以某一种计算机语言编程作为主线脉络，而是在一个个看似孤立的故事与工程中不断拾遗，并试着从中悟出一些道理。

本书适合大数据从业人员和对大数据相关知识感兴趣的人，初级和中级程序员、架构师及希望通过对数据的感知改进工作的人，产品经理、运营经理、数据分析师、数据库开发工程师等对数据分析工作敏感的人，以及所有对数据科学感兴趣并希望逐步深入了解数据科学知识体系的人阅读。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

数据科学家养成手册 / 高扬编著. —北京：电子工业出版社，2017.5
ISBN 978-7-121-31304-2
Ⅰ. ①数…　Ⅱ. ①高…　Ⅲ. ①数据管理—手册　Ⅳ. ①TP274-62

中国版本图书馆 CIP 数据核字（2017）第 071293 号

策划编辑：潘　昕
责任编辑：潘　昕
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
出版发行：电子工业出版社
　　　　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　　印张：23　　彩插：2　　字数：500 千字
版　　次：2017 年 5 月第 1 版
印　　次：2017 年 5 月第 1 次印刷
定　　价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

序

十几二十年前，读书是学习新技术的不二法门。当时如果要学习一门技术，都需要买上几本“砖头书”，一边阅读，一边动手，一页一页“啃”下来。很多在今天叱咤风云的高手，当年都是用这种方式打下基础的。

最近几年，技术学习的方式发生了深刻的变化，大量的在线视频课程、交互式学习环境、开箱即用的工具箱，使技术学习的效率大幅度提升，特别是在动手能力方面，培训效率有了质的飞跃。最近一年，受人工智能领域突破性进展的鼓舞，机器学习和数据科学成为技术圈中的显学，而在线学习成为主流的学习方式。在这种情况下，大批学习者仅仅看过一些视频教程，按要求在 Jupyter Notebook 中做过一些练习，就基本具备动手解决问题的能力，可以上岗了。

这当然很好。但是，倘若你想在某一个领域取得真知，读书仍然是不可或缺的手段。中国信息安全领域的领军人物冯登国院士曾经说，以他的经验，想要真的搞懂某一个领域，非得深入“啃”至少一本书不可。读书的效率相对于听课、看视频要低得多，而多维的知识体系通过单维的文字表达出来，也给理解带来了挑战。然而，唯其有这种挑战，才需要读者进入深度思考状态，使读书成为一个推敲、琢磨、设问和破解的过程。不经过这个过程，我们所学到的知识一般来说只能是浮于表面的，很难达到“知其然知其所以然”的高度。正因如此，我们已经开始发现，仅通过在线视频和动手练习的学习者，对于相关领域的理论掌握经常是肤浅的。可以说，到目前为止，读书作为一种学习手段，依然是其他方式无法取代的。

机器学习和数据科学领域有几本非常重要的著作，每一个有野心的学习者都应该选择至少一本深入研究。Christopher Bishop 于 2006 年出版的 *Pattern Recognition and Machine Learning*，Kevin Murphy 2012 年的巨著 *Machine Learning: A Probabilistic Perspective*，斯坦福大学两位机器学习泰斗 Trevor Hastie 和 Robert Tibshirani 及其学生合著的 *An Introduction to Statistical Learning*，当然还有 Ian Goodfellow 和 Yoshua Bengio 最近出版的 *Deep Learning*——称这几本书为这个领域的“四书五经”，应该没有争议。

但是，这几本书有一个共同的问题——都是按照教材的体例编写的，所以都是尽全力系统化地介绍知识，对这个领域丰富多彩的应用、历史、人文和故事却很少展开论述。而要成为一名数据科学家，仅有知识和动手能力是不够的，还需要有相应的素养，这包括特有的思维方式、

价值观，对相关历史背景和掌故的了解，以及对数据科学社区的认知和互动——这恰恰是本书的最大价值。

作者把数据科学放在一个更广阔的背景之中，从数学、统计学、方法论甚至认知论的层面出发，讨论数据科学的内涵和外延，内容丰富，旁征博引，语言生动，灵活有趣，帮助读者站在一个更丰富的势场中认识数据科学，理解数据科学的基本思想。尤为令人欣喜的是，作者将信息论、混沌理论纳入讨论之中，表明作者敏锐地注意到数据科学与系统科学和认识论的深层联系，这是难能可贵的。从这个角度来解说数据科学的书，应该说在中国是第一本，即使在全球范围内也是独具特色的。为此，我们愿意向读者推荐本书，并相信读者一定能从中获得非常有价值的启发。

CSDN、AI100 创始人　蒋　涛

AI100 合伙人　孟　岩

前　　言

为什么要写这本书

随着计算机科学和数据科学的发展，越来越多的人开始把目光投向其中最为耀眼的互联网、物联网、大数据、人工智能等高新技术领域，并且有相当多的高级技术人才已经在这些领域获得了令人瞩目的成就。

在追逐信息技术发展浪潮的过程中，数据科学成为人们在信息技术海洋中遨游所沉淀下来的理论与科学基础。我们都渴望通过对数据科学的理解来对生产工作进行指导和改善，这种工作的意义与其他各种在信息技术产业一线工作所创造价值的意义一样非比寻常。它给我们更广的辩证思考的空间，更高的观察事物的眼界，更多的自新的维度与动力。它是那么神秘且有趣。

今天，数据科学已经渗透到我们每个人的工作和生活之中。在你早上起来赶公车或者地铁的时候，你其实正在享受由数据科学辅助进行的精确调度服务；在你阅读工作报表的时候，你其实正在享受由数据科学辅助进行的大数据统计服务；在你吃午餐的时候，你其实正在享受由数据科学辅助进行的外卖快餐数据分发或食堂菜品改良服务；当你晚上回到家，在网上尽情购物的时候，你其实正在享受由数据科学辅助进行的高效电子商务和智能推荐服务。驾车出行有智能导航，就医问药有分诊机器人……也许你的家人或者朋友现在就在自己的工作岗位上，作为一名普通的销售人员、产品经理、人力资源师、售后服务人员、商务代表等，通过数据决策系统、数据库甚至电子表格来观察数据，作出判断，开展工作。数据科学给我们带来的红利已经紧紧把我们包围。

这本书绝无说教的想法，而是希望以书为媒，用谈天说地的方式，以激发每个人的思考为主要手段，归纳总结数据科学的实质及成就一位数据科学家所需要的基本素养。

遗憾的是，越是基础性、本源性的学科，与变成现实利益的距离就越远，让人觉得似乎不够实惠，不够亲近。至少读完这本书没办法帮你直接在第二天变出米饭、房子和汽车。不过我认为，楼房再高再漂亮，也需要人们看不到的深厚地基来支持；花儿再芬芳再娇艳，也需要在土壤之下吮吸养分的丰富根系来供能。这些看不到的东西，往往起着我们无法想象的巨大作用，

而这才是我希望与你一同讨论并思考的。

我们热爱生活，我们热爱所做的工作，我们希望在不断的攀登中看到更深更远的世界并去伪存真。那就让我们在点点滴滴的知识片段中一起开始慢慢思索、细细揣摩这一养成过程吧。

本书特色

本书从众多先贤及科学家的轶事开始讲起，以逐步归纳和递进的脉络总结出科学及数据科学应关注的要点，然后在生产的各个环节中对这些要点逐一进行讨论与落实，将这本书变成一本具有一定思维升华价值的参考书，从更高、更广的视角回看科学及数据科学在各个生产环节的缩影。

本书并不以高深的数学理论研究作为目的，也不以某一种计算机语言编程作为主线脉络，而是在一个个看似孤立的故事与工程中不断拾遗，并试着从中悟出一些道理。

简洁与深刻并重是本书的另一大特点。作为认知科学的延伸，数据科学一方面应该越来越引起广大大数据工作者的重视，另一方面也要撩开自己的神秘面纱，以最为亲民的姿态和每位大数据工作者成为亲密无间的战友，为用科学的思维方式进行工作做好理论准备。

读者对象

（1）大数据从业人员和对大数据相关知识感兴趣的人。

（2）初级和中级程序员、架构师，以及希望通过对数据的感知改进工作的人。

（3）产品经理、运营经理、数据分析师、数据库开发工程师等对数据分析工作敏感的人。

（4）希望在思维方式领域进行拓展的高校毕业生和希望接触并了解数据科学的社会人员。

（5）所有对数据科学感兴趣并希望逐步深入了解数据科学知识体系的人。

如何阅读本书

本书分为 3 篇，分别是认知篇、分化篇和实践篇。

认知篇（第 1 章 ~ 第 7 章）

归纳了什么是科学，数据科学的范围、定义与实践价值，以及辩证思维、哲学和实验的关系等问题。这些是认知观点的基石。

分化篇（第 8 章 ~ 第 11 章）

重点介绍了数据科学中与现代社会各行业联系最为紧密的统计学、信息论、算法学，另外把混沌论作为一个知识点进行了补充。这些是认知观点在不同细分学科中所形成的一些具体解决问题的思维方式和科学观点。

实践篇（第 12 章 ~ 第 19 章）

沿着数据生命周期进行演进。任何行业的数据生命周期都是按照采集、存储、统计与建模、算法、可视化与分析、决策支持的沿革来进行的，本篇对各个环节的注意事项和思维方式都做了详细的讨论，并在第 18 章介绍了两个具体的数据分析案例。

在本书的最后，补充了过去与同行们讨论过的，并在会议演讲及日常分享的过程中总结出来的一些精彩问答。

如果你希望读完这本书后能够在数学方面有很大的提升，在工程代码能力方面有巨大的进步，这本书恐怕帮不上什么大忙。但我相信，在读完这本书后，你会在一些以前并不熟知的领域有所了解和感悟，并逐步完善理解和分析问题的视角。如果你不是数据研究人员，也可以把这本书当成一个休闲读本。这本书里既没有太多的公式，也没有太过高深的理论，有的只是我在和你攀谈的过程中与你一起发现的新视角。

特别致谢

感谢绘麟社相辉先生和李晓林女士对本书的插画助力。

参加本书编写工作的有高扬、卫峥、左妍、尹会生、杨艺、陈钢、肖力。

勘误和支持

由于作者的水平有限，编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。如果您有更多的宝贵意见，欢迎扫描本页的二维码，关注“奇点大数据”微信公众号与我们进行互动讨论。本书后续的代码上传及勘误等相关更新内容都会在这个微信公众号发布。关注大数据尖端技术发展，关注奇点大数据。

同时，您也可以通过邮箱 77232517@qq.com 与我联系，期待能够得到您的真挚反馈，在技术之路上互勉共进。

高　扬

2017 年 1 月于珠海

目　录

认知篇

第 1 章　什么是科学家 …… 2

1.1　从太阳东升西落开始 …… 2
　1.1.1　农历 …… 2
　1.1.2　公历 …… 5
　1.1.3　小结 …… 7
1.2　阿基米德爱洗澡？ …… 7
1.3　托勒密的秘密 …… 10
1.4　牛顿为什么那么牛 …… 11
　1.4.1　苹果和三大定律 …… 11
　1.4.2　极限和微积分 …… 12
1.5　高斯——高，实在是高 …… 15
1.6　离经叛道的爱因斯坦 …… 17
1.7　本章小结 …… 20

第 2 章　什么是科学 …… 23

2.1　科学之科 …… 23
2.2　边界的迷茫 …… 23
2.3　科学之殇 …… 26
2.4　本章小结 …… 27

第 3 章　数据与数学 …… 28

3.1　什么是数据 …… 28
3.2　数学的奥妙 …… 29
　3.2.1　《几何原本》 …… 29
　3.2.2　《九章算术》 …… 30
　3.2.3　高等数学 …… 34
3.3　本章小结 …… 37

第 4 章　数据科学的使命 …… 38

4.1　走近数据科学 …… 38
　4.1.1　介质 …… 38
　4.1.2　从信息到数据 …… 41
　4.1.3　数据科学的本质 …… 43
4.2　万能的数据科学 …… 44
　4.2.1　测量 …… 44
　4.2.2　统计计算 …… 47
　4.2.3　指标 …… 52
4.3　使命必达 …… 53
　4.3.1　高效生产 …… 53
　4.3.2　破除迷信 …… 56
　4.3.3　目标一致与不一致 …… 57
4.4　本章小结 …… 58

第 5 章　矛盾的世界 …… 59

5.1　古希腊——学者高产的国度 …… 59

5.2 矛盾无处不在 61
5.3 世界究竟是否可知 63
5.4 薛定谔的“喵星人” 64
5.5 本章小结 67

第 6 章 实验和哲学 68

6.1 朴素的认知方法 68
6.1.1 眼见为实 69
6.1.2 归纳与总结 70
6.2 哲学靠谱吗 71
6.3 数学的尽头是哲学 72
6.4 本章小结 73

第 7 章 辩证思维 74

7.1 要不要辩证有多大区别 74
7.2 谁对谁错 76
7.3 做到客观不容易 77
7.4 观念的存弭 79
7.5 本章小结 82

分化篇

第 8 章 统计学 86

8.1 数理统计鼻祖——阿道夫·凯特勒 86
8.2 统计就是统共合计 88
8.3 数据来源 90
8.4 抽样 91
8.5 对照实验 91
8.6 误差 94
8.6.1 抽样误差 94
8.6.2 非抽样误差 96
8.7 概括性度量 97
8.7.1 集中趋势度量 98
8.7.2 离散程度度量 100
8.7.3 小结 100
8.8 概率与分布 100
8.8.1 数学期望 102
8.8.2 正态分布 103
8.8.3 其他分布 106
8.9 统计学与大数据 107

第 9 章 信息论 109

9.1 模拟信号 109
9.2 信息量与信息熵 110
9.3 香农公式 111
9.4 数字信号 112
9.5 编码与压缩 113
9.5.1 无损压缩 114
9.5.2 有损压缩 117
9.6 本章小结 126

第 10 章 混沌论 127

10.1 洛伦兹在想什么 128
10.2 罗伯特·梅的养鱼计划 129
10.3 有限的大脑，无限的维 130
10.4 “谋杀上帝”的拉普拉斯 132
10.5 庞加莱“不是省油的灯” 134
10.6 未知居然还能做预测 137
10.7 本章小结 137

第 11 章　算法学 ···· 139
11.1　离散的世界 ···· 139
11.2　成本的度量 ···· 142
11.3　穷举法——暴力破解 ···· 143
11.4　分治法——化繁为简 ···· 152
11.5　回溯法——能省则省 ···· 154
11.6　贪心法——局部最优 ···· 155
11.7　迭代法——步步逼近 ···· 156
11.7.1　牛顿法 ···· 157
11.7.2　梯度下降法 ···· 158
11.7.3　遗传算法 ···· 159
11.8　机器学习——自动归纳 ···· 161
11.8.1　非监督学习 ···· 162
11.8.2　监督学习 ···· 164
11.8.3　强化学习 ···· 175
11.9　神经网络——深度学习 ···· 177
11.9.1　神经元 ···· 177
11.9.2　BP 神经网络 ···· 180
11.9.3　损失函数 ···· 180
11.9.4　非线性分类 ···· 182
11.9.5　激励函数 ···· 186
11.9.6　卷积神经网络 ···· 188
11.9.7　循环神经网络 ···· 190
11.9.8　小结 ···· 193
11.10　本章小结 ···· 194

实践篇

第 12 章　数据采集 ···· 196
12.1　数据的源头 ···· 196
12.2　日志的收集 ···· 197
12.2.1　实时上传 ···· 197
12.2.2　延时上传 ···· 201
12.2.3　加密问题 ···· 202
12.2.4　压缩问题 ···· 203
12.2.5　连接方式 ···· 204
12.2.6　消息格式 ···· 206
12.2.7　维度分解 ···· 207
12.3　这只是不靠谱的开始 ···· 208
12.4　本章小结 ···· 209
第 13 章　数据存储 ···· 210
13.1　读写不对等 ···· 210
13.1.1　读多写少 ···· 211
13.1.2　读少写多 ···· 211
13.1.3　读写都多 ···· 212
13.2　进快还是出快 ···· 213
13.2.1　最快写入 ···· 213
13.2.2　读出最快 ···· 215
13.3　文件还是数据库 ···· 215
13.4　要不要支持事务 ···· 216
13.5　表分区和索引 ···· 218
13.5.1　表分区 ···· 219
13.5.2　索引 ···· 219
13.6　稳定最重要 ···· 222
13.7　安全性和副本 ···· 223
13.7.1　RAID ···· 223
13.7.2　软冗余 ···· 225
13.8　本章小结 ···· 226

第 14 章　数据统计 227

14.1　此“统计”恐非彼“统计” 227
14.2　要精确还是要简洁 231
14.3　统计是万能的吗 232
14.4　注意性能 234
14.5　本章小结 234

第 15 章　数据建模 235

15.1　模型是宝贵的财富 236
15.2　量化是关键 236
15.3　该算法出马了 237
15.3.1　统计学模型 238
15.3.2　线性关系 238
15.3.3　复杂的非线性关系 238
15.4　算法的哲学 240
15.5　本章小结 241

第 16 章　数据可视化与分析 242

16.1　看得见，摸得着 242
16.2　颜色很重要 242
16.3　别说布局没有用 244
16.3.1　由上而下，由简而繁 244
16.3.2　总–分，分–总，总–分–总 245
16.3.3　毗邻吸引 246
16.4　有图就别要表格 248
16.5　分析的内涵 249
16.5.1　相关性分析 250
16.5.2　预测分析 251
16.5.3　其他分析 252
16.6　有趣的统计应用 252
16.6.1　不规则图形的面积 252
16.6.2　套出你的实话 253
16.6.3　巧测圆周率 253
16.7　仁者见仁，智者见智 255
16.8　永恒的困惑 256
16.9　本章小结 257

第 17 章　数据决策 258

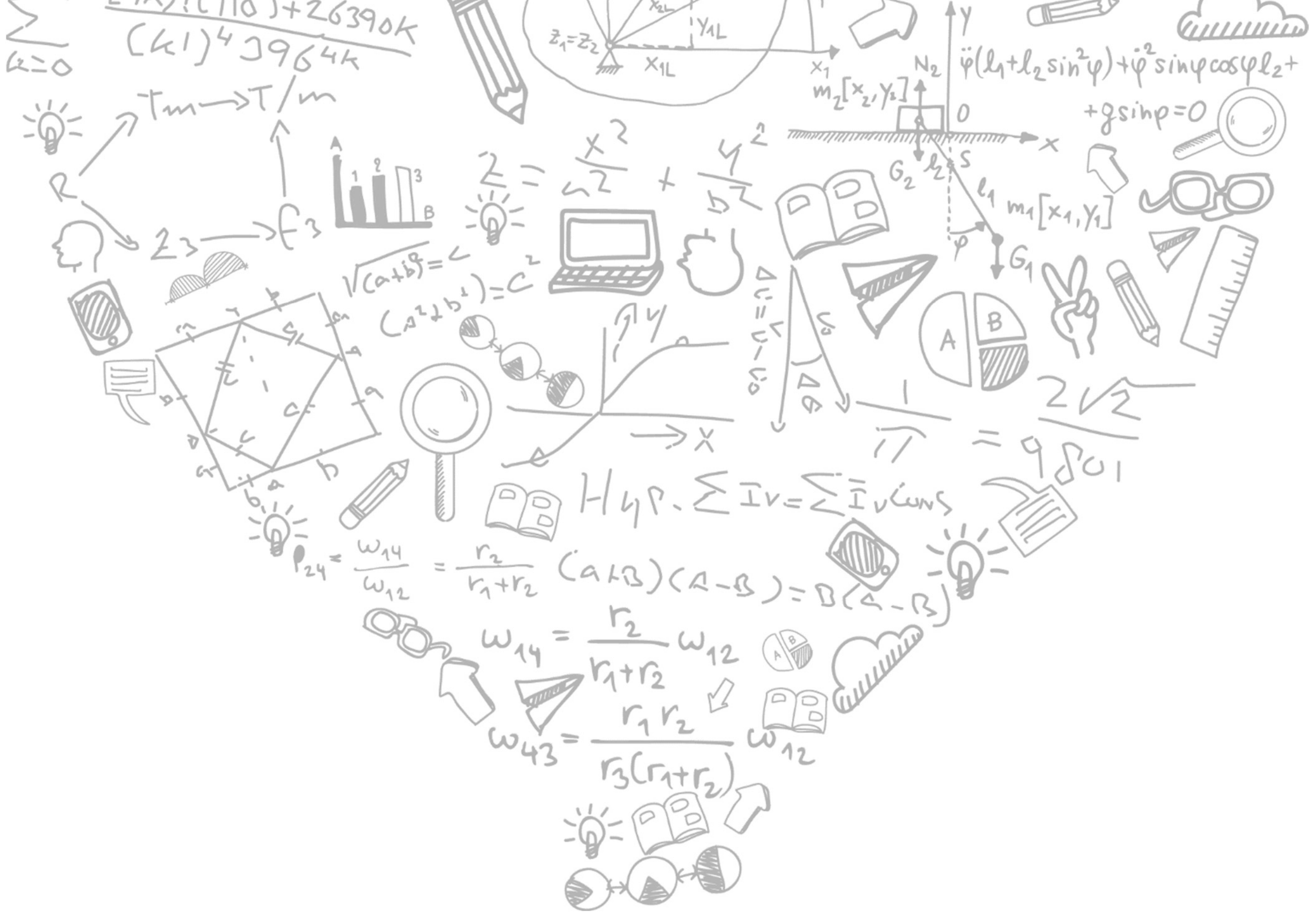
17.1　决策就是“拍脑袋” 258
17.2　哪里有物质，哪里就有数据 259
17.2.1　目的的统一 259
17.2.2　数据胜于雄辩 260
17.3　这是风险博弈 260
17.3.1　性价比优先 261
17.3.2　小迭代至上 262
17.3.3　不要“输不起” 262
17.3.4　留得青山在 263
17.4　本章小结 264

第 18 章　案例分析 266

18.1　K 线图里的秘密 266
18.1.1　什么是市场 267
18.1.2　谁在控制价格 267
18.1.3　货币价格的形成 270
18.1.4　零和博弈 271
18.1.5　涨跌都盈利 272
18.1.6　价格的预测 273
18.1.7　形态 274
18.1.8　K 线图周期 276
18.1.9　造市商与点差 277
18.1.10　科学分析 277
18.1.11　小结 310
18.2　数学能救命 310
18.2.1　阴云下的大西洋 311

18.2.2 护航船队的救星 ………… 311
18.2.3 数学家的天下 ………… 317
18.2.4 小结 ………… 317
18.3 人人都能运筹帷幄 ………… 317
第 19 章 与本书相关内容的问与答 ………… 319

后记 ………… 325
附录 A ………… 326
A.1 VMware Workstation 的安装 ………… 326
A.1.1 VMware 简介 ………… 326
A.1.2 安装准备工作 ………… 326
A.2 在 VMware 上安装 CentOS ………… 329
A.2.1 下载 ………… 329
A.2.2 安装 ………… 329
A.3 在 VMware 上安装 Ubuntu ………… 335
A.4 Python 语言简介 ………… 341
A.4.1 安装 Python ………… 341
A.4.2 Hello Python ………… 341
A.4.3 行与缩进 ………… 341
A.4.4 变量类型 ………… 342
A.4.5 循环语句 ………… 343
A.4.6 函数 ………… 344
A.4.7 模块 ………… 345
A.4.8 小结 ………… 345
A.5 Scikit-learn 库简介 ………… 346
A.6 安装 Theano ………… 346
A.7 安装 Keras ………… 347
A.8 安装 MySQL ………… 347
A.9 安装 MySQL-Python 驱动 ………… 349
A.10 MT4 平台简介 ………… 350
参考文献 ………… 353



认知篇

认知是人类与生俱来的能力，也是从蒙昧走向昌明的基本手段。在认知的过程中，人们去伪存真，在理性、科学地认识世界的道路上逐步强健起来。让我们沿着这条路回溯拾遗，看看前人给我们留下的宝贵遗产吧。

第 1 章　什么是科学家

从我们上学以前，就开始接受各种各样的启蒙教育，哪怕是捏橡皮泥、跳皮筋、玩弹球这些最普通、最不起眼的游戏项目，都能帮助我们锻炼协调性甚至认识世界。认识世界的过程贯穿我们每个人的一生。不仅如此，人类世世代代都在不断重复这个过程，日复一日，年复一年，从未停止。

纵览中国历史和世界历史，从古至今，各个时期都留下了很多故事。除了王侯将相们会留下他们的传记以外，还有两类人会留下他们的名字，而且这些名字会被人们自发记住并且广为传颂——一类是艺术家，人类的生命因他们的提炼而升华，人类的生活因他们的演绎而更加精彩；还有一类是科学家，他们一砖一木一石一瓦，为人类的进步搭建一级级坚实的台阶，他们是一群伟大的人。接下来我们要介绍的就是这群伟大的人。

在我们的印象中，科学家大都天资聪颖，智慧出众，在某一领域有深刻的理解。那么，科学家与常人都有哪些区别呢？是因为他们更聪明所以才能成为科学家吗？是因为他们记忆力殊绝于常人所以才能成为科学家吗？是因为他们比常人更加坚韧所以才能克服困难成为科学家吗？我认为，这些因素都只是成就科学家的一部分，而且既不是充分条件，也不是必要条件。

作为一个具有基本知识的常人，其实很多技术和科学原理我们都能理解而且加以应用。虽然有不少人能够纯熟地应用这些知识，但是在为数众多的常人中却很少有被称为“科学家”的，原因何在呢？

1.1　从太阳东升西落开始

1.1.1　农历

太阳东升西落，四季夏热冬冷，这种信息叫作常识。即使没有任何人来总结，但由于一个人每天、每月、每年都在经历这种交替，所以常人也可以总结出这些规律。总结出这些规律不足为奇，当然更不可能让每个人都因为对常识的总结或感知而成为科学家。

人们现在使用的历法，如公历、农历这种有完整的时间标识、日期标识、四季节气标识，并可以进行换算、时间加减、误差补偿的完整的历法体系，非常准确和精妙。中国人古代习惯使

用的农历，也叫“阴历”（这种叫法其实并不严谨，应该叫“阴阳历”）、“夏历”，从夏朝甚至更早以前就被人们总结出来并沿用至今，历朝历代都在为这个历法体系的健全进行修订。我们现在所用的农历历法体系大部分来自西汉太初年间修订的版本，也就是“太初历”[1]，所以称其为“汉历”可能更为合适。

在农历中，年、月、日换算的规则大致如下。

太阳东升西落算一日，月亮完整的一次盈亏周期（29.53059 天）算一月（朔望月）。地球公转和黄赤交角变化产生的四季轮回的平均时间为 365.2422 日，大约是一月周期的 12 倍。不过这样算一下，12 个朔望月和 1 个回归年的差距有 10.87512 天——不算小。

如果直接以 12 个朔望月作为一年，这样的结果在多年叠加后会让人感觉很滑稽。假设一年农历的正月初一和公历的 1 月 1 日在同一天——第 2 年公历 1 月 1 日的时候，农历已经是正月十一了；第 10 年公历 1 月 1 日的时候，农历已经是四月十八了——越往后越差得越多。

因为历法在很大程度上是为了对时间周期进行描述并指导农耕等生产行为而制定的，所以这种误差会使每个月对应的天气情况与地球公转的太阳年产生的真实情况有非常大的差别。如果要进行“几月播种、几月收割”这种知识的传授，几乎都要加一个“时间漂移”条件才能使用。而且，每隔 33.5851 年，以 12 个朔望月为一年的标准历法就要比太阳年历法少整整“1 年”。无论如何，这种历法体系都显得非常不健全。还好，先人们想出了“闰月”这个办法来进行误差补偿。

$$29.5306 \times i + 29.5306 \times 12 \times y = 365.2422 \times y$$

根据我们刚刚得到的条件，现在要求 i 和 y 的值，也就是说，在 y 个太阳年中加入 i 个闰月就能够把误差补齐。这样算出来，我们可以得到一组相对精确的数值：

$$y = 19，i = 7$$

带进去算一下，会得到误差为 0.0892 天，即在 19 个农历年中加入 7 个闰月就基本能够弥补 12 个朔望月所形成的一年和一个太阳年之间的大部分差值。0.0892 天的误差，约合 2.1408 小时。这种方法就是农历中的“十九年七闰月法”。

二十四节气（如表 1–1 所示）中的冬至是一个非常重要的节气。冬至是用圭表（如图 1–1 所示）观测标出的，这一天太阳直射南回归线。现在置闰的方法是：两个冬至之间，如果仅有 12 个朔望月则不置闰，如果有 13 个朔望月即置闰。置闰的月从冬至开始，当出现第一个没有中气的月份，这个月就是闰月，其名称是在前一个月的数字前面加一个“闰”字。例如，农历八月后需要置闰，那么八月后的这个月不叫“九月”，而叫“闰八月”，闰八月之后才是九月。

1　太初（前 104 年 ~ 前 101 年）是汉武帝的第七个年号。

表 1-1 二十四节气

季　　节	节气（节）	中气（气）
春	立春	雨水
春	惊蛰	春分
春	清明	谷雨
夏	立夏	小满
夏	芒种	夏至
夏	小暑	大暑
秋	立秋	处暑
秋	白露	秋分
秋	寒露	霜降
冬	立冬	小雪
冬	大雪	冬至
冬	小寒	大寒

图 1-1 圭表

有趣的是，农历中的正月、二月、三月到冬月、腊月，是按照朔望月来进行计算和标注的。但是，二十四节气却是根据对太阳运行周期规律的观测来进行标注和计算的，相当于对农历进行了补充——要不怎么说“农历”是“阴阳历”呢。所以，每一年的农历月和节气之间就会像和公历之间那样产生“错位”，农历月中就会有不太规律的缺少中气的月出现。不管是哪个月缺少中气，都要从冬至开始计算，对第一个没有中气的月做闰月处理。二十四节气最早被全数记录是在《淮南子·天文训》中，其中当然也包括对“十九闰七”这一天文现象的论述（如图 1-2 所示）。

二十四节气在中国民间已经扎根了几千年，我们依赖它进行农耕甚至基于中医药理论的人体调理，按照节气进行相应的补泻。这对“民以食为天”的中国人来说格外亲切，也有不少民间艺术家用充满灵性的画笔对这种古朴的文化进行了热情的演绎（如图 1-3 所示）。

图 1-2 淮南王刘安塑像和他的《淮南子》线装本

图 1-3 绘麟社手绘的节气与饮食

从“太阳东升西落，四季夏热冬冷”出发，就能完整地归纳出由太阳、地球、月球运行所产生的周期性规律，并能用它来指导农业生产，不能不说，这是一个非常重要并且有深远意义的科学体系。当然，和“太阳东升西落，四季夏热冬冷”相比，完整的农历就是科学的。虽然我们不知道最早开始制定农历的科学家是谁，但这不会只是一个人的成就，而是千百年来充满智慧的人通过不断观察、总结、实践、补充而逐渐形成的越来越完整的理论体系。这是一项平凡而伟大的工程。

1.1.2　公历

人们现在使用的公元纪年法（就是我们平常说的 1985 年、2016 年），全称“格里高利历”（Gregorian Calendar），这是公历的标准名称。

公元前 46 年，受到罗马共和国独裁官儒略·恺撒[2]的邀请，希腊数学家兼天文学家索西琴尼[3]（如图 1–4 所示）前往罗马，帮助创建精确的历法。经过精确的计算，罗马于公元前 45 年 1 月 1 日起执行了一种取代旧罗马历法的新历法，这种历法叫作“儒略历法”，简称“儒略历”（Julian Calendar）。

图 1–4　索西琴尼

儒略历规定：一年划分为 12 个月，大小月交替；每 4 年一闰，平年 365 日，闰年 366 日（在当年 2 月底增加 1 个闰日）；年平均长度为 365.25 日。

但是，由于太阳年的长度是 365.2422 日，这样每年的误差为 0.0078 日，每 400 年误差就达到 3.12 日——这个误差就比较大了。到公元 1500 年左右，儒略历的误差超过了 11 天。1582 年 3 月 1 日，教皇格里高利十三世颁布了以儒略历为基础并进行了改善的格里高利历，即沿用至今的公历。

公历的主要规定如下。

第一，1582 年 10 月 4 日后的一天是 10 月 15 日，而不是 10 月 5 日，但星期序号仍然连续计算，即 10 月 4 日是星期四，10 月 15 日（当年 10 月 4 日之后的第一天）是星期五。这样，就把从儒略历颁布到当时积累的误差天数一笔勾销了。

第二，为避免再发生春分“漂移”的现象，将公元年数能被 4 整除的年作为闰年，而当公元年数最后 2 位均为“0”（即“世纪年”）时，能被 400 整除的年才是闰年。这样，误差进一步缩小了。

格里高利历的历年平均长度为 365 日 5 时 49 分 12 秒，仅比回归年长 26 秒。虽然照此计算，大约 3000 年后仍存在 1 天的误差，但这样的精确度已经相当了不起了。

2　即凯撒大帝。盖厄斯·儒略·凯撒，公元前 102 年 7 月 ~ 公元前 44 年 3 月。

3　索西琴尼（Sosigenes，生卒年代不详），古希腊数学家、天文学家。

到了现代，我们有了更多先进的计时方式，也能够测算由于潮汐、地震、行星撞击等多种地球自身地质因素及天文因素所造成的时间误差，甚至出现了“闰秒”等特殊调整方法。

1972 年，国际计量大会决定，当世界时与原子时之间相差超过 0.9 秒时，就在协调世界时上加上或减去 1 秒，以尽量接近世界时，这就是闰秒。1972 年以来实施闰秒的时间如表 1–2 所示。我国研制的铯原子钟如图 1–5 所示[4]。

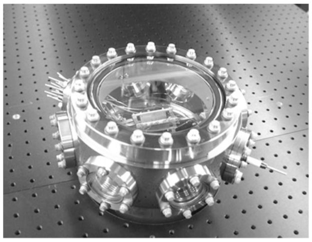


图 1–5　我国研制的铯原子钟

表 1–2　1972 年以来实施闰秒的时间

年　份	6 月 30 日 23:59:60	12 月 31 日 23:59:60	年　份	6 月 30 日 23:59:60	12 月 31 日 23:59:60
1972	+1 秒	+1 秒	1989		+1 秒
1973		+1 秒	1990		+1 秒
1974		+1 秒	1992	+1 秒	
1975		+1 秒	1993	+1 秒	
1976		+1 秒	1994	+1 秒	
1977		+1 秒	1995		+1 秒
1978		+1 秒	1997	+1 秒	
1979		+1 秒	1998		+1 秒
1981	+1 秒		2005		+1 秒
1982	+1 秒		2008		+1 秒
1983	+1 秒		2012	+1 秒	
1985	+1 秒		2015	+1 秒	
1987		+1 秒	—	—	—

是否加入闰秒由位于巴黎的国际地球自转和参考坐标系统服务（International Earth Rotation Service，IERS）决定，在格里高利历的每年 6 月或 12 月的最后一天的最后一分钟进行跳秒或不跳秒处理，也就是说，每年的这两个“1 分钟”不一定是 60 秒。

如果是正闰秒，则这一秒是加在第二天 00:00:00 之前的。当决定加入正闰秒的时候，当天 23:59:59 的下一秒记为 23:59:60，然后才是第二天的 00:00:00。如果是负闰秒，当天 23:59:58 的下一秒就是第二天的 00:00:00 了。虽然有这样的设计，不过到目前为止还没出现负闰秒的情况。

不设置闰秒会有什么影响呢？按照世界时与原子时之间时差的累积速度来看，39 年会减慢 24 秒，这样在 10000 年后，世界时与原子时之间的时差就是 1.7094 小时，我们看到的现象就是：

4　中国计量科学研究院研制并运行的“NIM5 铯原子喷泉钟”，2000 万年误差小于 1 秒。

在不加闰秒的情况下，公元 12016 年再夸青少年是“早上八九点钟的太阳”就不太合适了，因为这个时候太阳有可能还没升起来。

1.1.3　小结

从简单的“太阳东升西落，四季夏热冬冷”就能总结出如此完备的历法体系，并能够指导人们在几千年的文明史中进行周而复始的、极有规律的作息和生产生活，对农业、天文、地理、历史等人类自然科学和社会科学的记述与改善提供极大的帮助，这是一件多么有意义的事情!

科学渗透在人类生活的每个角落，对任何事物，都不要觉得它渺小甚至无足重轻。科学有一个非常平易近人的特性就是“以小见大”——再小的事情，只要认真研究和发掘，都能不断升华，最后变成一个庞大的、完整的、精密的体系结构。我们在本书中还能看到很多这样的例子。

1.2　阿基米德爱洗澡?

从小到大，我们在很多儿童读物、科普材料甚至教科书里都见过关于阿基米德（公元前 287 年 ~ 公元前 212 年）洗澡的图画（如图 1–6 所示）。

图 1–6　阿基米德在洗澡[5]

不同尺寸，不同风格，惟妙惟肖，以至于幼年时期的我一度误以为阿基米德是一位爱洗澡的老人家。在各种版本的科普材料里，都有阿基米德发现阿基米德原理的故事。

浸入静止流体中的物体受到一个浮力，其大小等于该物体所排开的流体质量，方向垂直向上并通过所排开流体的形心（如图 1–7 所示）。该原理是阿基米德首先提出的，故称“阿基米德原理”。这一结论不仅对部分浸入液体中的物体同样适用，还可以推广到气体。

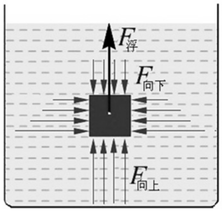


图 1–7　物体在液体中受到的浮力

完整的故事是这样的。相传叙拉古赫农王（King Hieron）让工匠替他做了一顶纯金的王冠，当然，黄金都是叙拉古赫农王作为材料交给工匠的。金冠做好后，虽然质量与叙拉古赫农王当初交给工匠的相同，但叙拉古赫农王还是心下狐疑——他怀疑工匠私吞黄金，在金冠里掺

5　图片来自百度图库。

了假。叙拉古赫农王希望能有一个方法来检验金冠是否是纯金的，但又不想破坏王冠。最后，决定请阿基米德来检验皇冠的纯度。

最初，阿基米德对这个问题也是无计可施。直到有一天，他在家洗澡，当坐进澡盆时，看到水往外溢，突然想到可以用测定固体在水中排水量的办法来确定金冠的体积。他兴奋地跳出澡盆，连衣服都顾不上穿就跑了出去，大喊："尤里卡！尤里卡！"（ $\varepsilon\rho\eta\kappa\alpha$ ，意思是"找到了"。）

经过进一步实验，阿基米德来到王宫。他把王冠和同等质量的纯金分别放在两个盛满水的盆子里，比较两个盆子里溢出来的水，发现放王冠的盆子里溢出来的水比另一个盆子多。这就说明，王冠的体积比相同质量的纯金的体积大，也就是说，王冠的密度比纯金小，证明王冠里掺进了其他金属。

$$m = \rho v$$

m 是质量，ρ 是密度，v 是体积。m 一定的时候，ρ 和 v 成反比。

这次实验的意义远大于给私吞黄金的人定罪。阿基米德从中发现了浮力定律（也叫阿基米德原理），即物体在液体中所获得的浮力，等于它所排出液体的质量，也就是广为人知的排水法。直到现在，中学物理课本的力学部分还有关于浮力计算的课程，这都归功于阿基米德的重要发现。

当然，阿基米德作为哲学家、数学家、物理学家，贡献远远不只是发现阿基米德原理。著名的杠杆原理也是他发现的（如图 1–8 所示）。

$$F_1 \times L_1 = F_2 \times L_2$$

F_1 是杠杆一侧的受力大小，L_1 是与 F_1 同侧的杠杆长度；F_2 是杠杆另一侧的受力大小，L_2 是与 F_2 同侧的杠杆长度。

伴随着这个伟大的原理，还有一个让人如雷贯耳的句子："给我一个支点，我可以撬起地球。"这是阿基米德在发现杠杆原理后，在极为激动的情况下给叙拉古国王希伦（Heron）写信时写下的一句"豪言壮语"（如图 1–9 所示）。其实，仔细想想就会知道，即使是在科技昌明的今天，要想实现这句话所描述的情景也仅有理论上的可能。地球的质量约为 6×10^{24} 千克，假设一个人所能产生的推力是 100 千克[6]（力气已经不算小了），那么杠杆两侧的长度比应该是 6×10^{22} 比 1。也就是说，即使在地球这一侧杠杆仅为 1 米长，在阿基米德那一侧也需要一根 6×10^{22} 米长的杠杆才行。从支点移动到杠杆的边缘去撬动杠杆，即便用光速[7]前往，也需要约 2×10^{14} 秒，也就是约 6341958 年的时间。所以，说这是一句颇具浪漫主义风格的"豪言壮语"绝非对阿基米德的诋毁。不过，没有人会从文学角度用这句话跟一位 2000 多年前的科学家较真，因为我们从中获得的精神鼓舞远比真正实现这个实验的受益大得多。

6　2016 年里约奥运会男子举重 69 公斤级比赛冠军——中国举重运动员石智勇的抓举成绩也仅为 162 千克。

7　约为 3×10^{8}m/s。

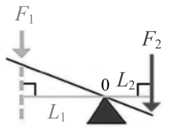


图 1–8　杠杆原理

图 1–9　雕塑作品——阿基米德翘起地球

除此之外，阿基米德还尝试使用穷竭法（逼近法）来解决曲面面积问题，也就是用内接和外切的直边图形不断地逼近曲边形的方法，例如求圆的面积——在阿基米德的《论球和圆柱》中有清楚的记载（如图 1–10 所示）。

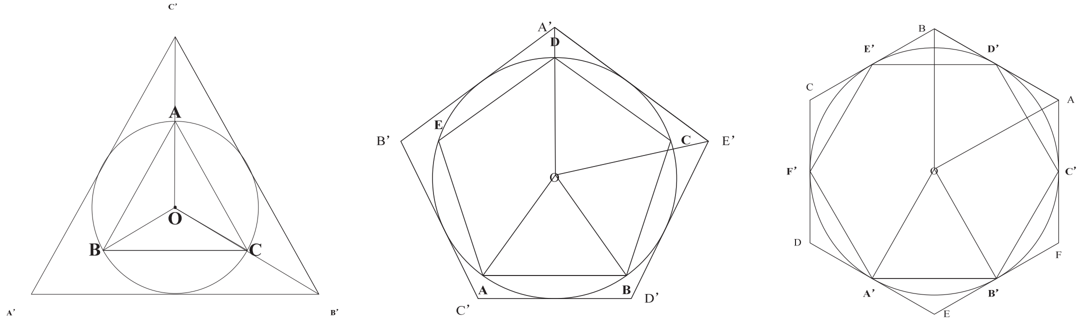


图 1–10　多边形内接圆和外切圆

可以用圆的内接三角形和外切三角形，内接四边形和外切四边形，内接五边形和外切五边形……一直向下分割。这样，内侧的多边形和外侧的多边形都能够用三角形组合的方式进行面积加和，而每种情况下，圆形的面积就在内外两个多边形的面积之间。

$$S_{inner} < S_{round} < S_{outer}$$

然后，通过求 S_{inner} 和 S_{outer} 的极限逼近 S_{round} 的值。这种思路在高等数学的“夹逼定理”[8] 中也有体现，精妙至极！而且，这种方法还能根据公式

$$S = \pi r^2$$

求出 π 的近似值。在丈量手段落后的 2000 多年前，欧洲已经有阿基米德这样的聪明人发明出如此精妙的办法来测量圆周率，不能不说是个创举。

可以肯定的是，阿基米德的很多成就也是基于前人的成果继续进行总结与发扬得到的，例如欧几里得[9]（如图 1–11 所示）。

图 1–11　欧几里得

8　Sandwich Theorem，由法国著名数学家约瑟夫–路易斯·拉格朗日（Joseph–Louis Lagrange，1736 年 1 月 ~ 1813 年 4 月）提出。

9　欧几里得（公元前 330 年 ~ 公元前 275 年），古希腊著名数学家，有“几何之父”的美誉。

阿基米德之所以能成为世人心目中的科学家，并不是因为他在洗澡的时候发现水会对他有浮力，而是因为他能用精确的方法测定浮力的大小，以及浮力和排出水的体积的具体换算关系。使用杠杆会省力，估计是当时的建筑工人广泛应用的起重技巧，但以精确的方式对杠杆两侧受力平衡时的力矩与力的大小进行关系换算，仍旧是阿基米德第一个想出来并记载下来的。这是他与世人的不同，也是他成就中最为重要的因素。

1.3 托勒密的秘密

如果托勒密这个名字你不熟悉，“地心说”你估计不会陌生。托勒密就是“地心说”的集大成者。

克罗狄斯·托勒密（Claudius Ptolemaeus，约 90 年 ~ 168 年，如图 1–12 所示），古希腊天文学家、地理学家、占星学家和光学家。他著有 4 本重要著作，分别是《天文学大成》（*Almagest*）、《地理学》（*Geography*）、《天文集》（*Tetrabiblos*）和《光学》（*Optics*）。

图 1–12 克罗狄斯·托勒密

其中，《天文学大成》这本书受到世人的赞誉当属最多。这本书总结了当时近 500 年的希腊天文学和宇宙学思想，并统治天文学界长达 13 个世纪，直到 15 世纪才被哥白尼的“日心说”推翻。在托勒密的认知体系下，假设宇宙是以地球为中心的，所有天体以均匀的速度按圆形的轨道绕转。

我们在中学物理课上就已经学过，实际上，“天体的运动以变速按椭圆轨道绕地球以外的各天体中心运动”的描述最为简洁。例如，可以描述地球围绕太阳公转就是在一个椭圆轨道上，而太阳就在这个椭圆的其中一个焦点上。然而，托勒密认知的局限性显然没有让他认识到这一点。毫无疑问，这种假设与观测到的天体运动特性也有差距。

所以，要想维护原来的基本假设，就要考虑用一些非常复杂的几何形状来进一步做补充解释。托勒密使用了“秘密武器”——3 种复杂的原始设想——本轮、偏心圆和均轮（如图 1–13 所示）。在加入这些辅助模型后，他能对火星、金星和水星等的轨道分别给出合理的描述。但是，如果把它们放在一个模型中，尺度和周期将发生冲突。而且，这种模型的问题还是刚才说的——描述起来太复杂。

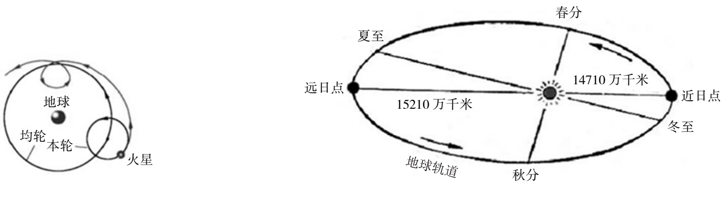


图 1–13 托勒密的均轮和本轮模型

说到这里，也许有些读者朋友会觉得托勒密也没什么了不起，写了半天还写了个错误的理论。可是，这种理论在当时的场景局限下也是一种可以被印证的理论，不然也不可能成为传世之作，而且一传就是 13 个世纪。至于后来教会对“日心说”的提出者——尼古拉·哥白尼（1473 年 2 月 ~ 1543 年 5 月）的迫害，也跟托勒密没什么直接关系，毕竟罗马天主教廷是因为“日心说”违反《圣经》才这样做的。

托勒密的成就在于自己建立了一套相对完备的、自圆其说的天文运行系统，而且是在单筒望远镜发明前 1000 多年[10]就已经进行了相对精确的定性与量化，演算模型也相对完整，这在当时实属不易。客观地说，“日心说”也好，“地心说”也好，只要整个体系描述不矛盾就行，把哪个天体当成这个“不动”的中心参考点都可以，区别只在于对描述其他天体运行状态时的复杂度不同。想想看，是不是这样？

1.4　牛顿为什么那么牛

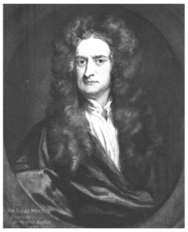


图 1–14　艾萨克·牛顿

艾萨克·牛顿（Isaac Newton，1643 年 1 月 ~ 1727 年 3 月，如图 1–14 所示）爵士，英国皇家学会会长，英国著名的物理学家，百科全书式的全才，著有《自然哲学的数学原理》（*Philosophiae Naturalis Principia Mathematica*）、《光学》等。

1.4.1　苹果和三大定律

图 1–15　牛顿的苹果落地

我们每个人都很熟悉牛顿的名字，估计大部分人第一次听说他，都是因为那个被苹果砸了脑袋之后“脑洞大开”，提出万有引力的故事（如图 1–15 所示）。近年来，也有人说这个故事不过是杜撰而已。

不管牛顿和苹果的故事是真是假，让牛顿名垂青史的终究是他一生的巅峰产物——牛顿三大定律和微积分。

牛顿三大定律也叫作牛顿运动定律，内容如下。

（1）牛顿第一运动定律：孤立质点保持静止或做匀速直线运动。

$$\sum_{i=1}^{n} F_i = \frac{dv}{dt} = 0$$

10　1608 年荷兰人李波尔赛造出了世界上第一架望远镜，随后由意大利数学家、物理学家、天文学家伽利略加以改造和发扬。

F_i是质点受到的各个力，都是矢量；$\sum_{i=1}^{n} F_i$是指合力；v是运动速度，是矢量；t是时间；$\frac{dv}{dt}$是v对t求导，也就是速度的变化率。速度变化率为0，表示要么静止，要么是匀速直线运动。

（2）牛顿第二运动定律：动量为p的质点，在外力F的作用下，其动量随时间的变化率同该质点所受的外力成正比，并与外力的方向相同。

$$F = \frac{dp}{dt}$$

p和F都是矢量。我们平时应用更多的是

$$F = ma \quad 或者 \quad a = \frac{F}{m}$$

两者从数学关系上的区别仅仅是“在$a = \frac{F}{m}$中m不能为0”——在经典物理力学的范畴中，质量m也确实不可能为0，但是$a = \frac{F}{m}$从因果关系上对“加速度a是由外力F和质量m来决定”的解释更为直观一些。

$$p = mv$$

所以，化简得到

$$F = \frac{dp}{dt} = \frac{mdv}{dt} = ma$$

（3）牛顿第三运动定律：相互作用的两个质点之间的作用力和反作用力总是大小相等，方向相反，作用在同一条直线上。

$$F_{12} = -F_{21}$$

F_{12}和F_{21}都是矢量，负号仅仅表示两者方向相反，F_{12}表示质点2受到的质点1的作用力，F_{21}表示质点1受到的质点2的作用力。

牛顿三大定律是我们在中学物理课上的必学内容，也是量化宏观低速物理中最为经典的力学内容。不管当初牛顿是不是因为被苹果砸了脑袋才想出了三大定律，他的研究发现终究成为后世机械、航空、制造等诸多现代化工业领域的基石，可谓功不可没。

1.4.2 极限和微积分

除了牛顿三大定律，在大学理工科专业必学的微积分同样与牛顿的理论研究有着非常重要的关系。

从17世纪开始，随着欧洲社会的进步和生产力的发展，以及航海、天文等许多领域有大量新问题需要解决，数学也开始研究变化量，进入了“变量数学”时代。在17世纪有数十位科学家为微积分的创立进行了开创性的研究，但使微积分逐步体系化并成为数学的一个重要分支的

还是牛顿和莱布尼茨[11]这两位数学家。在前面我们提到的阿基米德计算圆面积的过程中已经包含了这种“从有穷到无穷”的逼近思想，这或许就是微积分思想最为原始的萌芽吧。在那个时代，不仅在西欧，在中国同样有智者在用类似的方式计算圆的面积。

3 世纪中期，魏晋时期的数学家刘徽（约公元 225 年 ~ 295 年，如图 1–16 所示）首创割圆术，为计算圆周率建立了严密的理论和完善的算法。所谓割圆术，就是不断倍增圆内接正多边形的边数求出圆周率的方法（如图 1–17 所示）。

图 1–16　刘徽

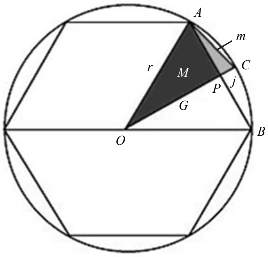


图 1–17　刘徽割圆术示意图

根据刘徽的记载，此前人们求证圆面积公式时，是用圆内接正十二边形的面积来代替圆面积。应用“出入相补”原理，将圆内接正十二边形拼补成一个长方形，借用长方形的面积公式来论证《九章算术》中的圆面积公式。刘徽指出，这个长方形是以圆内接正六边形周长的一半作为长，以圆半径作为高的长方形，它的面积是圆内接正十二边形的面积。这种“合径率一而弧周率三也”（即后来常说的“周三径一”）的论证当然不严密。他认为，圆内接正多边形的面积与圆面积都有一个差，用有限次数的分割、拼补无法证明《九章算术》中所述的圆面积公式。因此，刘徽大胆地将极限思想和无穷小分割引入数学证明。他从圆内接正六边形开始割圆，“割之弥细，所失弥少，割之又割，以至不可割，则与圆周合体，而无所失矣”。也就是说，将圆内接正多边形的边数不断加倍，则它们与圆面积的差就越来越小，而当边数不能再增加的时候，圆内接正多边形面积的极限就是圆面积。

刘徽考察了内接多边形的面积，也就是它的“幂”，同时提出了“差幂”的概念。“差幂”是后一次与前一次割圆的差值，可以用图 1–17 中小灰色三角形面积的倍数来表示。刘徽指出，在用圆内接正多边形逼近圆面积的过程中，圆半径在正多边形与圆之间有一段余径。以余径乘以正多边形的边长，即 2 倍的“差幂”，加到这个正多边形上，其面积则大于圆面积。这是圆面积的一个上界序列。他认为，当圆内接正多边形与圆是合体的极限状态时，“则表无余径。表无余径，则幂不外出矣”。也就是说，余径消失了，余径的长方形也就不存在了，因此圆面积的这个

11　戈特弗里德·威廉·莱布尼茨（Gottfried Wilhelm Leibniz，1646 年 7 月 ~ 1716 年 11 月），德国犹太族哲学家、数学家，被誉为“17 世纪的亚里士多德”。

上界序列的极限也是圆面积。于是，内外两侧序列都趋向于同一数值，即圆面积。

微积分（Calculus）是高等数学中研究函数的微分（Differentiation）、积分（Integration）及有关概念和应用的数学分支，是数学的一个基础学科。微积分的内容主要包括极限、微分学、积分学及其应用等。微分学包括求导数的运算，是一套关于变化率的理论。它使函数、速度、加速度和曲线的斜率等均可用一套通用的符号进行讨论。积分学包括求积分的一系列运算，为定义和计算面积、体积等提供了一套通用的方法。

牛顿在 1671 年写了《流数术和无穷级数》(*Methodus Fluxionum Et Serierum Infinitarum*）一书，但这本书直到 1736 年才出版。牛顿在这本书里指出，变量是由点、线、面的连续运动产生的，否定了以前他认为的变量是无穷小元素的静止集合。牛顿把连续变量称为流动量，把这些流动量的导数称为流数。牛顿在流数术中所提出的中心问题是：已知连续运动的路径，求给定时刻的速度（微分法）；已知运动的速度，求给定时间内经过的路程（积分法）。

戈特弗里德·威廉·莱布尼茨（如图 1–18 所示）也是为微积分理论的创立立下不朽功勋的大数学家。其实，牛顿和莱布尼茨的研究是独立进行的，并在相近的时间先后完成。比较特殊的是，牛顿创立微积分要比莱布尼茨早 10 年左右，但正式公开发表微积分这一理论，莱布尼茨却要比牛顿早 3 年。他们的研究各有长短。在那时，由于民族偏见，关于微积分发明优先权的争论竟从 1699 年开始延续了 100 多年。

图 1–18　戈特弗里德·威廉·莱布尼茨

$$g(x)=\int f(x)dx$$

我们现在一直使用的积分符号 $\int$ 其实是莱布尼茨创造的。

（1）定积分

如果 $f(x)$ 在 $[a, b]$ 上连续，并且存在原函数 $F(x)$，那么

$$\int_a^b f(x)dx=F(b)-F(a)=F(x)|_a^b$$

也可以写作

$$\int_a^b F'(x)dx=F(b)-F(a)$$

这个公式也叫作“牛顿–莱布尼茨”公式，是定积分含义的公式体现。

（2）微分

设函数 $y=f(x)$ 在某区间内有定义，x_0 及 $x_0+\Delta x$，在此区间内如果增量

$$\Delta y=f(x_0+\Delta x)+f(x_0)$$

可以表示为

$$\Delta y = A\Delta x + o(\Delta x)$$

其中，A 是不依赖 Δx 的常数，那么称函数 $y=f(x)$ 在点 x_0 是可微的，而 $A\Delta x$ 叫作函数 $y=f(x)$ 在点 x_0 相应于自变量增量 Δx 的微分，记作 dy，即

$$dy = A\Delta x$$

以一元函数 $f(x)$ 为例，设 $y=f(x)$，有

$$f'(x)=\lim_{x\to x_0}\frac{f(x)-f(x_0)}{x-x_0}$$

$f'(x)$ 就是 $f(x)$ 的导函数，也称作导数。

$$dy = df(x)=f'(x)dx$$

也可以记作

$$\frac{dy}{dx}=\frac{df(x)}{dx}=f'(x)$$

在牛顿和莱布尼茨总结成型的微积分理论中，这样的思维方式贯彻始终，是一种完美的对极限和“从有穷到无穷”的思维方式的定量阐释。微积分学科的创立，对近现代高精尖工业的发展有着奠基性的伟大作用。没有微积分就不可能制造出高性能的精密机床、先进的舰船、舒适节能的汽车，也不可能研究出能以超过 300 千米的时速安全行驶的高速列车，更别说装配航空发动机、大型客机和航天运载火箭了。没有微积分，今天的世界一定不会这么缤纷多彩。

1.5　高斯——高，实在是高

在我很小的时候读过一套叫作《十万个为什么》[12] 的科普丛书，非常有趣。在这套书的数学卷中提到过下面这个小故事。

高斯 10 岁的时候进入学校里教授算术的班。这是一个首次创办的班，孩子们在这之前都没听说过“算术”这门课程。班上的数学教师是布特纳，他对高斯的成长也起了一定的作用。

一天，布特纳老师布置了一道题：1+2+3……这样从 1 一直加到 100 等于多少？当大家还在一步步计算的时候，高斯很快就算出了答案——5050。起初，布特纳老师并不相信高斯算出了正确答案，他说：“你一定是算错了，回去再算算。”高斯是这样算的：1+100=101，2+99=101，3+98=101……以此类推，从 1 到 100 中有 50 组这样的数，所以答案很快就能计算出来——50×101=5050。这个故事数百年来被传为佳话，说明高斯在很小的时候就有极为出众的数学计算技巧。

12　这里指的是 1980 年由少年儿童出版社出版的《十万个为什么》第四版。

高斯的全名是约翰·卡尔·弗里德里希·高斯（Johann Karl Friedrich Gauss，1777 年 4 月 ~ 1855 年 2 月），他是德国著名的数学家、物理学家、天文学家、大地测量学家，近代数学的奠基者之一，也被认为是历史上最重要的数学家之一。高斯一生成就极为丰硕，以他的名字命名的成果达 110 个，可谓“前无古人，后无来者”，对数论、代数、统计、分析、微分几何、大地测量学、地球物理学、力学、静电学、天文学、矩阵理论和光学皆有贡献。高斯的主要著作如表 1–3 所示。

表 1–3　高斯的主要著作

出版时间	著　　作
1801 年	《算术研究》
1809 年	《天体运动理论》
1827 年	《曲面的一般研究》
1843 年或 1844 年	《高等大地测量学理论》（上）
1846 年或 1847 年	《高等大地测量学理论》（下）
1839 年	《地磁的一般理论》
1840 年	《地磁概念》
1840 年	《论与距离平方成反比的引力和斥力的普遍定律》

在大学学习统计学的过程中，我们一定会接触正态分布（Normal Distribution）这个名词。正态分布也叫“高斯分布”（Gaussian Distribution），公式如下。

若随机变量 x 服从一个位置参数为 μ、尺度参数为 σ 的概率分布，其概率密度函数为

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

正态分布的概念是由法国裔英国籍数学家亚伯拉罕·棣莫弗[13]（如图 1–19 所示）于 1733 年用阶乘的近似公式导出正态分布概率曲线时首次提出的，不过由于高斯率先将这一公式应用于天文学领域的研究并广泛使用，所以后世还是将这一公式所表示的分布特点称为“高斯分布”。

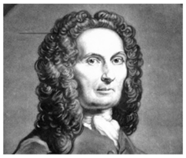


图 1–19　亚伯拉罕·棣莫弗

高斯使用的“最小二乘法”发表于他 1809 年的著作《天体运动论》中。1829 年，高斯提供了最小二乘法的优化效果强于其他方法的证明，该证明被称为“高斯–马尔可夫定理”（Gauss – Markov Theory）。这种方法后来长期应用于线性回归算法中的残差分析，直到今天。

高斯一生发表了多达 155 篇高质量的论文。他对待学问的态度十分严谨，只把他自己认为

13　亚伯拉罕·棣莫弗（Abraham De Moivre，1667 年 5 月 ~ 1754 年 11 月），法国数学家。

十分成熟的作品发表出来。后世对高斯的赞誉极高。贝尔[14]曾经这样评论高斯："在高斯死后，人们才知道他早就预见到一些 19 世纪的数学知识和领域，而且在 1800 年之前已经期待它们的出现。如果他能把他所知道的一些东西公布出来，很可能比当今数学还要先进半个世纪或更长的时间。"爱因斯坦也曾高度评价高斯："高斯对近代物理学的发展，尤其是对相对论的数学基础所作的贡献（主要指曲面论），其重要性是超越一切，无与伦比的。"

从 1989 年到 2001 年年底，高斯的肖像和他所写的正态分布曲线，以及一些在哥廷根（高斯去世的地方）非常突出的建筑物，都一直在 10 德国马克的钞票票面上出现。此外，德国发行了 3 种纪念和表彰高斯的邮票。一种邮票（第 725 号）发行于 1955 年，即高斯逝世 100 周年；另外两种邮票（第 1246 号和第 1811 号）发行于 1977 年，即高斯诞辰 200 周年（如图 1–20 所示）。

图 1–20　邮票上的高斯像

高斯对近现代数学的贡献，我们从贝尔和爱因斯坦这两位大科学家的评价中可见一斑。他对数学的贡献，尤其是对数学基础研究的贡献，为后来一系列数学分支领域的深入研究奠定了坚实的基础。

1.6　离经叛道的爱因斯坦

阿尔伯特·爱因斯坦（1879 年 3 月 ~ 1955 年 4 月，如图 1–21 所示），著名的犹太裔物理学家。1879 年出生于德国乌尔姆市，1900 年毕业于瑞士苏黎世联邦理工学院，1901 年加入瑞士国籍。1905 年获苏黎世大学哲学博士学位，提出光子假设，成功解释了光电效应。不过在 16 年后的 1921 年，爱因斯坦才因此获得诺贝尔物理学奖，同年又创立狭义相对论。狭义相对论依据两条原理：相对性原理和光速不变原理。1915 年，爱因斯坦提出了广义相对论引力方程的完整形式，并成功解释了水星近日点运动。

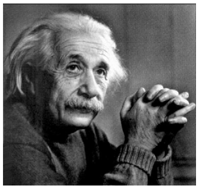


图 1–21　阿尔伯特·爱因斯坦

爱因斯坦留下了一个几乎世人都能知晓但大都一知半解的公式——

$$E = mc^2$$

这个公式就是著名的爱因斯坦质能方程，它揭示了物质质量与能量的关系。该公式表明物体相对于一个参照系静止时仍然有能量，这与牛顿经典力学中对机械能的阐述是有矛盾的。所以，在这个理论刚刚提出的时候，没有人看得明白，更别说深刻理解了。这个公式还有多种表

14　亚历山大·格雷厄姆·贝尔（AlexanderGrahamBell，1847 年 ~ 1922 年），英国裔美国发明家和企业家。他发明了世界上第一台可用的电话机，创建了贝尔电话公司，被世界誉为"电话之父"。

现形式，其中之一是

$$\Delta E = \Delta M_v c^2$$

根据这个公式，又可以推导出质速关系（质速关系当然也可以由其他方式推导出来），也就是时间会随速度增加而变慢，质量变大，长度减小。

当外力作用在静止质量为 m_0 的自由质点上时，质点每经历位移 ds，其动能增量为 $dE_k = Fds$。设外力作用于质点的时间为 dt，则质点在外力冲量 Fdt 的作用下，其动量增量是 $dp = Fdt$，而这一时刻的 $v = \frac{ds}{dt}$，则可以推导出质点的速度表达式为 $v = \frac{dE_k}{dp}$，也就是

$$dE_k = vdp = vd(mv) = v^2dm + mvdv$$

而在这个相对允许运动的关系参考系中进行坐标变化就需要使用洛伦兹变换，所以质量变换公式变成了 $m = \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$，去根号化简得到 $m^2(c^2 - v^2) = m_0^2c^2$，对速度 v 求导。

$$\frac{d(m^2(c^2 - v^2))}{dv} = \frac{d(m_0c)^2}{dv} = 0$$

$$m^2\frac{d(c^2 - v^2)}{dv} + \frac{dm^2}{dv}(c^2 - v^2) = 0$$

$$mvdv = (c^2 - v^2)dm$$

代入 $dE_k = v^2dm + mvdv$，可以得到 $dE_k = c^2dm$。

当 $v = 0$，质量 $m = m_0$，动能 $E_k = 0$，对 $dE_k = c^2dm$ 做定积分。

$$E_k = \int_{m_0}^{m} c^2dm = mc^2 - m_0c^2$$

在物质质量从 m_0 变化到 m 时，物质释放出来的能量就是 $m_0 - m_0$ 的值乘以 c^2，这是一个非常大的数字。在现代核工业领域，就是基于这样的计算原理对核反应释放能量的大小进行核算。

物理学上有一个名词叫作湮灭（Annihilation），指的是物质与它的反物质相遇时会发生完全性的物质到能量的转换过程。在这个过程中，$m = 0$，释放出来的能量就是物质静止状态的质量 m_0。以 0.5g 物质和 0.5g 反物质湮灭为例，以近似值 $c = 3\times10^8$ 来计算，湮灭将释放出 9×10^{16} 焦耳的能量。

我们拿这个值和三峡电站[15]的发电能力做一个对比。2015 年，三峡电站发电量为 870 亿千瓦时[16]，也就是约 3.1×10^7 焦耳的能量。这么巨大的能量只需要 3.48 克的物质发生湮灭就可以释

15　中国电网的中心支点，总装机容量达 2250 万千瓦，是世界最大的水电站和清洁能源基地。

16　信息来源于《中国能源报》。

放出来！这也是美国、德国等很多发达国家的尖端实验室不断对其进行研究的重要原因之一。

原子核中蕴含的大量能量被发现后，在和平利用能源方面得到了较好的应用——建造核电站。截至 2014 年，世界主要有核国家的核电在本国电力工业中的能源贡献都占据了相当高的比例。其中最高的是法国，年核电发电量 407438 千瓦时，发电量世界排名第 2 位，但是占比为第 1 位，为 74.8%，也就是说，全法国的发电站每发 100 度电，就有 74.8 度是核电站贡献的，这个比例不能不让人惊讶。美国的年核电发电量为 770719 千瓦时，发电量世界排名第 1 位，占比 19%，列世界第 16 位。我国经过多年的建设，年核电发电量也上升至 92652 千瓦时，世界排名第 6 位，不过占比较低，仅有 2%，列世界第 30 位。

不过，科学是一柄双刃剑。在过去的 40 年里，发生过三哩岛核电事故（1979 年，美国）、切尔诺贝利核电事故（1986 年，前苏联，如图 1–22 所示）及福岛核电事故（2011 年，日本），这些重大的事故曾多次引起人们的核恐慌。尽管如此，从人类发展的总趋势上仍旧不能阻止核电作为新型清洁能源逐步取代火电和水电的历史走向。

核能在刚刚被人们发现的时候可不是用来发电和支持工业生产的，而是被用来制造杀人的武器。1945 年，也就是第二次世界大战的最后一年，美国制造了 3 枚原子弹，还给它们起了外号——“小男孩”、“瘦子”和“胖子”。1945 年 7 月 16 日，美军在新泽西州阿拉摩哥多进行了代号为“瘦子”的原子弹试验并获得了成功，旋即将其投入军事用途。

1945 年 8 月 6 日，美军在日本广岛投下了“小男孩”——一枚只有 4 吨重但设计当量高达 2 万吨 TNT 的原子弹，当天造成 8.8 万人死亡，5.1 万人受伤或失踪，其中包括一定比例的日本军人（如图 1–23 所示）。同年 8 月 9 日，“胖子”降临日本长崎。2.2 万吨设计当量的“胖子”当天造成 10 万多人的伤亡和失踪。

图 1–22　切尔诺贝利核电站周边被荒废的游乐场

图 1–23　核爆后的广岛一片焦土

原子弹爆炸在世界上产生了巨大的反响，同时也产生了两方面声音的对话。一方面，饱受战争迫害的人们对原子弹爆炸造成的破坏，在解了心头恨之余，更多的是惊骇，尤其是在战争过后，冷静下来，也觉得这样一种拥有极高杀伤力的武器可能会给人类的安全造成隐患，这种

顾虑在“冷战”[17]期间尤甚；另一方面，从一开始就关注核能科学的科学家们纷纷表示，使用这种技术来做杀人的武器，有悖于研究此项技术的初衷。爱因斯坦在晚年对自己当初提议建造原子弹的想法颇感后悔，他曾坦言：“早知如此，我宁可当个修表匠。”[18]这也许就是爱因斯坦在最终从“潘多拉宝盒”中放出核能这只巨兽时仅能表示的感叹了。

爱因斯坦的理论有着划时代的意义。从此，人们从原子的外部“进入”原子的内部，这在以前是不可想象的。而且，由于爱因斯坦的理论太过超前，以至于当时世界上最顶尖的科学家对他的研究也是不知所云，这直接导致 1905 年提出的光电效应理论直到 1921 年才被科学界普遍接受并获得诺贝尔物理学奖。

1.7 本章小结

“科学家”的称谓或者说评价其实是比较苛刻的，需要在相关领域有超人的建树才能够被大家公认为是科学家（尤其是大科学家）。那么，具备什么性质的人才能称为科学家呢？通过几位科学家的事迹，我们可以做一个总结。

（1）突破性

对加、减、乘、除这些运算，即使一个人通过学习可以熟练掌握，但充其量也就是“熟练工”——不能称为“科学家”。而从加、减、乘、除运算生发出来的微积分理论则是一种全新的概念，它虽然基于加、减、乘、除的运算基础，但是具有自己非常独特的应用优势与技法，并因此完成了对众多复杂领域的描述和计算——这种突破性可以成就伟大的科学家。

（2）精确性

太阳东升西落，四季夏热冬冷，这种事情不足为奇。通过观察星辰运动，得出这些天体都在围绕地球转动（至少看上去是这样），也不值得称道。而能够把这一切通过量化的（尤其是精确量化的）描述进行阐释和计算的人，确实比常人的观察层面更广、更深。这种精确描述的过程与状态的现实指导意义远比从大方向上指出一个性质要大。

（3）体系性

体系性就是指“自圆其说”的框架。所谓“自圆其说”的概念是指一种科学理论要有完整的对象定义，以及对象之间的换算与转化的条件和量化值的确切关系，通常表现为各种各样的计算公式或者统计概率的确切描述。而且，这一切有一个前提——需要“可应验”，或者严格地说是“高度应验”才行。

这种体系性应该是所有性质里最为重要的一点，因为体系框架的成立会形成一个封闭的讨

17 冷战（Cold War）是指 1946 年至 1991 年，以美国、北大西洋公约组织为主的资本主义阵营与以苏联、华沙条约组织为主的社会主义阵营之间的政治、经济、军事斗争。

18 一说为爱因斯坦在惊闻原子弹爆炸于广岛和长崎后所发出的懊悔的感叹。

论边界。在讨论边界内部的所有观察对象都会以各种量化结果呈现在换算公式的因子里，它的完整性会直接支持体系本身的存在，并支持所有在该体系下的性状判断与预测。

我认为，在理论上具备突破性、精确性、体系性这 3 点才能被称为“大科学家”。再看看其他科学家的事迹，大部分也都符合这 3 点。一个科学家或者有素养的学者，其研究最起码要具备精确性和体系性。从重要度上来说，体系性大于精确性，精确性大于突破性。为什么呢？

首先，体系性是一种对完整的、彼此印证的关系描述，而不是一个孤立的观点。孤立的观点或者事件是个案性的，难以总结出规律，也缺乏泛化性。换句话说，偶然性的东西在我们将其研究并整理成系统的理论之前是不能被轻易称为“科学”的。

“初，至和元年五月，晨出东方，守天关，昼见如太白，芒。”[19]这是北宋钦天监对公元 1054 年 7 月 4 日天宫星附近超新星爆发事件的记录。后来据天文学家推断，金牛座蟹状星云形成的起源就是这次超新星爆炸（如图 1–24 所示）。虽然这次记录时间准确，内容也比较翔实，但要说这个记录是一种科学成果，我觉得太过牵强。它只是一个以内容描述为主的客观的观测记录，而关于超新星的成因、变化条件，以及是否存在其他超新星、这些超新星的变化是否对周边天体有影响、影响是什么这一系列的论述，都没能提出。如果把这种仅仅停留在纯粹的观察结果层面的描述记录算作科学成果，就有些不严肃了。在这里，我并不是全盘否认中国古代天文学的成就，只是说把一次孤立的天文现象观测记录算作天文学发现有失严谨。

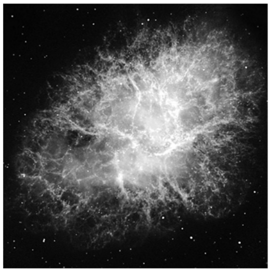


图 1–24　金牛座蟹状星云

其次，我们来说精确性。精确性会给人准确的数字依据，从而进行精确的判断或预测。这种判断是一个从性质到数量的变化，同样也是精确度的变化。有些内容是个案性的，可能在短时间内不能成就一个完整的体系，但是由于精度的提升，同样能够成就一位科学家或者至少帮他实现一个方面的成就。水上可以行船，这不是什么高深的知识，但是要得出水上的船承载多少货物而不会倾覆，就需要超出当时常人认知水平的研究能力了。同样，杠杆的使用也许早就存在，但是杠杆两侧受力对比的具体数值，也是当时常人不曾了解的知识。而这些阿基米德都做到了，这也是他众多成就中的一部分。

最后，我们来说说突破性。其实突破性可以说是蕴含在精确性和体系性之中的，理论体系的形成之于零散的知识点和实验结果就是一种突破，精确数值描述之于性质样态的描述在精度层面也是一种突破。除此之外，某些假说的提出，某些新观点的阐释，某些新领域的尝试，也是科学家做出成就的地方。有不少科学家只是提出了某种假说或者理论，但是由于种种原因——不论是观测能力的原因，还是其他客观条件的限制等——没有成型的理论体系产生，即使当时不

19　出自《宋会要》。

被世人所理解或赞同，但仍然瑕不掩瑜，为后世所赞许。

在德米特里·伊万诺维奇·门捷列夫[20]（如图 1–25 所示）整理出相对完整的元素周期表（如图 1–26 所示）之前，英国的分析化学家和工业化学家约翰·亚历山大·雷纳·纽兰兹（John Alexander Reina Newlands，1837 年 ~ 1898 年）已经发现了化学元素的性状具备周期性。1865 年，纽兰兹把当时已知的 61 种元素按原子量递增的顺序排列，发现每隔 7 种元素便出现性质相似的元素，如同音乐中的音阶一样，因此称为“元素八音律”（如图 1–27 所示）。

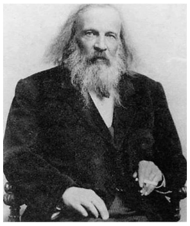


图 1–25　德米特里·伊万诺维奇·门捷列夫

ПЕРИОДИЧЕСКАЯ СИСТЕМА ЭЛЕМЕНТОВ

ГРУППЫ ЭЛЕМЕНТОВ

		I	II	III	IV	V	VI	VII	VIII	0
1	I	H 1 1,008								He 2 4,003
2	II	Li 3 6,940	Be 4 9,02	5 B 10,82	6 C 12,010	7 N 14,008	8 O 16,000	9 F 19,00		Ne 10 20,183
3	III	Na 11 22,997	Mg 12 24,32	13 Al 26,97	14 Si 28,06	15 P 30,98	16 S 32,06	17 Cl 35,457		Ar 18 39,944
4	IV	K 19 39,096	Ca 20 40,08	Sc 21 45,10	Ti 22 47,90	V 23 50,95	Cr 24 52,01	Mn 25 54,93	Fe 26 55,85; Co 27 58,94; Ni 28 58,69	
	V	29 Cu 63,57	30 Zn 65,38	31 Ga 69,72	32 Ge 72,60	33 As 74,91	34 Se 78,96	35 Br 79,916		Kr 36 83,7
5	VI	Rb 37 85,48	Sr 38 87,63	Y 39 88,92	Zr 40 91,22	Nb 41 92,91	Mo 42 95,95	Ma 43 —	Ru 44 101,7; Rh 45 102,91; Pd 46 106,7	
	VII	47 Ag 107,88	48 Cd 112,41	49 In 114,76	50 Sn 118,70	51 Sb 121,76	52 Te 127,61	53 J 126,92		Xe 54 131,3
6	VIII	Cs 55 132,91	Ba 56 137,36	La 57 * 138,92	Hf 72 178,6	Ta 73 180,88	W 74 183,92	Re 75 186,31	Os 76 190,2; Ir 77 193,1; Pt 78 195,23	
	IX	79 Au 197,2	80 Hg 200,61	81 Tl 204,39	82 Pb 207,21	83 Bi 209,00	84 Po 210	85 —		Rn 86 222
7	X	87 —	Ra 88 226,05	Ac 89 227	Th 90 232,12	Pa 91 231	U 92 238,07			

* ЛАНТАНИДЫ 58-71

Ce 58 140,13	Pr 59 140,92	Nd 60 144,27	61 —	Sm 62 150,43	Eu 63 152,0	Gd 64 156,9
Tb 65 [illegible]	Dy 66 162,46	Ho 67 164,94	Er 68 167,2	Tu 69 169,4	Yb 70 173,04	Cp 71 174,99

图 1–26　门捷列夫的元素周期表

No.		No.		No.		No.		No.		No.		No.		No.	
H	1	F	8	Cl	15	Co&Ni	22	Br	29	Pd	36	I	42	Pt&Ir	50
Li	2	Na	9	K	16	Cu	23	Rb	30	Ag	37	Cs	44	Os	51
G	3	Mg	10	Ca	17	Zn	24	Sr	31	Cd	38	Ba&V	45	Hg	52
Bo	4	Al	11	Cr	19	Y	25	Ce&La	33	U	40	Ta	46	Tl	53
C	5	Si	12	Ti	18	In	26	Zr	32	Sn	39	W	47	Pb	54
N	6	P	13	Mn	20	As	27	Di&Mo	34	Sb	41	Nb	48	Bi	55
O	7	S	14	Fe	21	Se	28	Ro&Ru	35	Te	43	Au	49	Th	56

图 1–27　纽兰兹的元素八音律

纽兰兹的这个想法当时一是由于太过超前，二是因为不是一种完整的理论体系，所以未被人们广泛接受。直到元素周期体系确立后，人们才承认他的重要发现。不过，他仍然获得了英国皇家学会的褒奖——1887 年，学会颁发给他一枚戴维奖章。

我们对比纽兰兹和门捷列夫的周期表，会发现两者的差别还是很大的，尤其是门捷列夫的元素周期表中对第八族元素的归类，以及根据原子外层电子数和化合价对元素进行的划分，吻合度是很高的，元素族的考虑也更为科学。但是，纽兰兹同样是一位值得后人尊重的化学科学家，因为他的尝试开创了一种崭新的视角，会给后人的研究带来很大的提示和重要的线索。

突破性、精确性、体系性的递增是一种“人无我有，人有我优，人有我精”的层次变化，不管是在哪个科学领域，万变不离其宗。这也是科学家的事迹激励我们前进的重要动力。

20　德米特里·伊万诺维奇·门捷列夫（1834 年 2 月 ~ 1907 年 2 月），俄国科学家。

第 2 章　什么是科学

2.1　科学之科

中国古汉语体系中多为单字词，所以大家也能猜得出，“科学”一词是个舶来的词汇。明治维新时期日本学界初用“科学”一词翻译英文中的“Science”，这个词是由日本启蒙思想家西周（にし　あまね，1829 年 ~ 1897 年，如图 2-1 所示）翻译并引入的（科学——かがく）。在欧洲的语言体系中，该词来源于拉丁文“Scientia”，意为“知识”、“学问”，在近代侧重关于自然的学问。

图 2-1　西周

科，会意字，“从禾从斗，斗者量也”，所以“科学”一词取义“测量的学问”。这里所说的“测量的学问”本身就有一种精确性的含义，再上到“学问”的范畴，就带有比较严格的推敲色彩了，而经得住推敲的东西通常体系性比较强，能够相互支撑，自圆其说。这就是“科学”这个词的字面含义。

科学是一种泛指的领域，包括数据科学、计算机科学、社会科学、经济科学、语言科学、生物科学等，这些各领域的科学有各自的体系、规范及认知特性。这些都是小、窄而深的认知领域。科学，不管是某个领域的科学，还是泛指的科学认知体系，都有最基本的要求——精确性和体系性，即使在突破性上的特点不明显，也必须保证其精确性和体系性的存在和可靠。

2.2　边界的迷茫

人类在不断认知世界的过程中成长起来。随着科学水平的进步，观测手段的强化，工具技术的发展，人们对眼镜、耳朵、四肢都做了极大程度的“延伸”。

人的肉眼在天气晴好的情况下可以在海平面上看到距离自己 20 ~ 30 千米的船只（视船体大小），在高山顶上可以看到距离自己超过 300 千米的城市建筑轮廓。在使用望远镜的情况下，远处的物体通常可以放大 8 ~ 10 倍，这样，观察距离自己 20 千米的物体就像观察距离自己 2 千米

的物体一样。在我国贵州省建设的500米口径球面射电望远镜（FAST）[21]如图2-2所示。

图 2-2　FAST 球面射电望远镜

使用这种大型射电望远镜可以观测130亿光年以外的星体。不过，也有人说这种观测方式的被动性比较强，与其说是“看”，不如说是“听”。这种观测方式，在距离上的提升是极大的，从20千米的距离提升到130亿光年，大约提升了6.15×10^{13}倍。

人类不断扩展自己视野的边界，不断认识新天体、新星系，也不断发掘新的天文现象，并逐步推翻现有的天文学认知，建立新的天文学知识体系。在任何一个历史时期，人类既无法说清现在所掌握的知识有哪些局限，也无法预测未来会有什么新的发现可以补充现有的知识，这就是认知边界扩展过程中一直困扰着人类的事情。

望远技术的提升不断带给我们新的发现，形成新的知识颠覆，而对不同电磁波长的感知也让人类在认知体系上有了质的飞跃（如图2-3所示）。

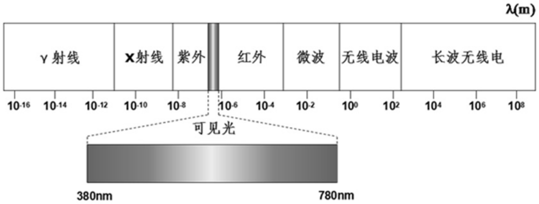


图 2-3　电磁波长分布

人们最初只能用肉眼看到可见光部分的电磁波，也就是波长为380～780nm的电磁波。对光的波长从大到小粗略划分，就是我们平时说的“红橙黄绿青蓝紫”七色光。

21　在本书出版之前，此望远镜已经投入使用了。

波长比红色可见光还要大的电磁波称为红外线。红外线是在 1800 年由英国天文学家、古典作曲家、音乐家弗里德里希·威廉·赫歇尔（Friedrich Wilhelm Herschel，1738 年 11 月 ~ 1822 年 8 月，如图 2–4 所示）通过棱镜分光发现的。此后，由于物体的红外辐射现象，人类又发明了很多应用性很强的工业产品，例如夜视仪（如图 2–5 所示）。

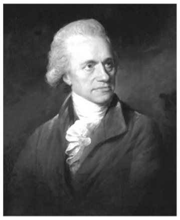


图 2–4　弗里德里希·威廉·赫歇尔

图 2–5　使用军用红外夜视仪看到的画面

红外线在医学上也有非常好的临床应用效果。红外线中对人体益处最大的就是 4 ~ 14μm（4000 ~ 14000nm）这个波段范围。因为这个红外线波段对生命的生长有促进作用，能够活化细胞组织，促进血液循环，提高人的免疫力，加强人体的新陈代谢，所以在医学界统称为“生育光线”。家用医疗产品中形形色色的红外治疗仪，大都是利用红外线的这种性质。

波长比紫色可见光还要小的电磁波叫作紫外线。紫外线是电磁波谱中波长 10 ~ 400nm 辐射的总称。1801 年，德国物理学家、化学家里特（Johann Wilhelm Ritter，1776 年 ~ 1810 年）在日光光谱的紫端外侧发现了一段能够使含有溴化银的照相底片感光的电磁波，从而发现了紫外线。

随着观测技术水平的发展，波长更大的微波、无线电波、长波，以及波长更小的 X 射线和 γ 射线也逐渐被发现。

从在可见光的认知范围内不断徘徊，到 20 世纪 30 年代后期发现超长波（波长 $1 \sim 10^7$m）存在的这个漫长的探索过程中，没有人能预测下一步或者在什么时候、什么情况下我们能发现新的波长形式的电磁波。尽管我们可以猜测可能会有某种波长的电磁波存在，但我们通常无法预言观测的具体条件、时间和地点。这种对科学未来的无法预知性不仅存在于光学这个小小的领域，在物理学的其他领域，以及化学、地理学、天文学、数学等领域，我们同样有着对边界扩展的迷茫，甚至在很多时候，我们难辨方向。

美国前国防部长唐纳德·亨利·拉姆斯菲尔德（Donald Henry Rumsfeld，1932 年 7 月 ~ ）在他的回忆录《已知与未知》中写过一段堪比绕口令的话：“据我们所知，有‘已知的已知’，有些事，我们知道我们知道；我们也知道，有‘已知的未知’，也就是说，有些事，我们现在知道我们不知道。但是，同样存在‘未知的未知’——有些事，我们不知道我们不知道。”

应该说，这段话的本意不是指认知科学中的迷茫，但是放在这里却有异曲同工之妙——确实，我们在对客观世界的认知过程中会时常伴随这种“已知的已知”、“已知的未知”、“未知的

已知”和“未知的未知”。这让我们头痛不已，但同时也让我们对在不断开拓认知边界的过程中所产生的乐趣充满了无穷的期待。

2.3 科学之殇

科学的迭代和进化有的时候是量的变化，有的时候是质的变化。如果说量的变化能够让我们欣喜不已，那么质的变化通常会让我们热血沸腾，因为这是开启新纪元的钥匙。

然而，不是所有的科学新纪元都是和平到来的。相反，科学新纪元的到来通常都带着或多或少的伤感，或者伴随着令人难以承受的代价。

公元前 212 年，古罗马军队入侵叙拉古。阿基米德并没有因为是一位伟大的数学家而获得任何优待，他被一名古罗马士兵杀死，终年 75 岁。对此最感惋惜的是古罗马军队的统帅马塞拉斯，他将杀死阿基米德的士兵当成杀人犯处决，还为阿基米德举行了隆重的葬礼。此外，马塞拉斯在西西里岛为阿基米德修建了一座陵墓并将他葬在那里。根据阿基米德的遗愿，墓碑上刻有“圆柱内切球”这一几何图形。一代数学巨人竟死于战乱中一个无知小卒之手，这也令后世许多科学家唏嘘不已。

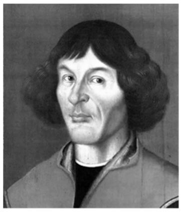


图 2-6 尼古拉·哥白尼

尼古拉·哥白尼（1473 年 ~ 1543 年，如图 2-6 所示）是文艺复兴时期波兰天文学家、数学家。在 40 岁时，他提出了“日心说”。虽然“日心说”仍旧摆脱不了当时的观测条件给人带来的认知局限，但是“日心说”带来的长期而深远的影响还是让世人震惊。“日心说”从一定意义上否定了教会的权威，改变了人类对自然和自身的看法，以一种更简洁、更优美的方式阐释了天体运动的关系。当时罗马天主教廷认为“日心说”违反《圣经》，但哥白尼仍坚信“日心说”，并认为“日心说”与《圣经》不存在矛盾。哥白尼经过长年的观察和计算，完成了他的伟大著作《天体运行论》。

1533 年，60 岁的哥白尼在罗马进行了一系列的讲演。但是迫于教皇势力的压制，他临近古稀之年才终于决定将《天体运行论》出版。1543 年 5 月 24 日，病势深沉的哥白尼在卧榻上见到了从纽伦堡寄来的《天体运行论》样书，他只摸了摸书的封面，便再没有力气一页页翻阅，并于两天后——1543 年 5 月 26 日辞世。

爱因斯坦同样是一位伟大的物理学家。由于他的思想过于超前，以至于他提出的“狭义相对论”和“广义相对论”在当时引起一片哗然，而且直到今天仍然不是每个人都能理解的概念。他在 1905 年提出的“光子假设”和“光电效应”的概念，也由于科学界的认知问题、政治因素、种族因素等，直到 1921 年才获得了迟到 16 年之久的诺贝尔物理学奖。这公平吗？我并不这么认为。可是，要成为一位为世人认可的伟大的科学家，这个门槛是一定要自己迈过去的。

玛丽·居里（Marie Skłodowska Curie，1867 年～1934 年，即“居里夫人”，如图 2-7 所示）是法国著名波兰裔科学家、物理学家、化学家。1903 年，居里夫妇和法国物理学家安东尼·亨利·贝克勒尔（Antoine Henri Becquerel，1852 年～1908 年）由于对放射性的研究共同获得诺贝尔物理学奖。1911 年，她因发现元素钋和镭，再次获得诺贝尔化学奖，成为历史上第一个两获诺贝尔奖的人。居里夫人的成就包括开创放射性理论、发明分离放射性同位素技术、发现元素钋和镭。在她的指导下，人们第一次将放射性同位素用于治疗癌症。但是，由于长期接触放射性物质，居里夫人于 1934 年 7 月 3 日因恶性白血病逝世，而且在晚年饱受放射性后遗症的折磨，痛苦不堪。

图 2-7　居里夫人

在人类探索科学、攀登科学高峰的过程中，成名者是少数中的少数，大多数人都被淹没在历史的滚滚尘埃中，成为整座历史高墙中的一块小小的砖头，甚至只是填缝的泥土。而成名者在这个过程中同样会遇到各种难以想象的困难——资源的缺乏，实验条件的限制，成果不被世人或者科学界认可，人格不被尊重，积劳成疾的痛楚等。只有那些对科学有着崇高信仰的人，只有那些能够耐得住寂寞并且对自己研究的事物有着坚定信心的人，才有可能最终摘得科学的桂冠。这其中的代价与酸甜苦辣只有科学工作者自己知道。这些伟大的科学家，永远值得世人敬重与缅怀！

2.4　本章小结

在对科学有了完整的认识以后，就可以开始了解科学在各个领域的渗透了。至于一种理论或一种学说是不是科学，我们已经可以通过最基本的定义来初步判断。不过，要从头整理一套经得起推敲的科学理论并非易事，很多时候，不仅可能要消耗一代甚至几代科学家毕生的心血，还需要科学家具有耐得住寂寞的平和性格和坚强不屈的伟大信仰。

这种科学的信仰和理念是一种可遵循的路径，是保证我们将前进的航船驶向正确方向的指南针。所以，要想研究数据科学，就一定要对科学有足够的认识。只有这样，我们才不会被充斥在周围的乱象所迷惑，最终获得正确的认知观念和知识。

第 3 章　数据与数学

3.1　什么是数据

数据的英文是“data”，其实这是复数形式的“数据”，其单数形式是“datum”——英文解释是“a fact or piece of information”，即一个事实或一条信息。那么，从字面去解释“data”，就是很多事实或信息的混合体。

数据是什么？这几乎成为一个我们熟视无睹的问题。

我们平时看到的数字和字符、字母，都有可能是数据。

先看下面一组例子（如图 3–1 所示）。这里有 6 个 0，请问它是数据吗？

000000

图 3–1　例子（1）

再看一组例子（如图 3–2 所示）。这里有 4 个 1 和 2 个 a，请问它是数据吗？

1111 aa

图 3–2　例子（2）

我们假设这两个例子都有一些比较特殊的前置场景。假设第一组的 6 个 0 其实是时、分、秒的简写，“000000”表示 00 点 00 分 00 秒，“112349”则表示 11 点 23 分 49 秒，那么它是不是数据呢？假设第二组出现的 4 个 1 和 2 个 a 其实是一组密码，“1111”代表一个约定地点，“aa”代表一个约定事件，这组数字和字母的意义就有了相应的解读，那么它是不是数据呢？

符号如果想被认定为数据，就必须承载一定的信息。信息很可能因场景而定，因解读者的认知而定，所以，符号是否能作为数据使用，解读者的主观视角具有决定性作用。解读者是否能够正确解读信息，将直接导致相应的符号是否能够被认定为数据。

数据无处不在。在网络上、报纸上、书籍上，在家用电脑的磁盘上，在我们的脑海里，数据大量且广泛地存在。围绕数据，人类已经做了很多数据科学的研究和应用工作，其中最为基

础的就是数据的存储和传输。数据的价值在于数据上所承载的信息，信息的价值在于其消除不确定性时的成本及其直接和间接的作用，而这一系列的价值使得我们对数据的价值予以空前的重视，这就是我们热衷于研究数据科学的原因所在。

3.2　数学的奥妙

3.2.1　《几何原本》

数学充满神奇的魅力，是一门研究数量、结构、变化、空间及信息等概念的学科，是一门抽象且普适的基础学科，是超越万物的不同性质而追求相同性质（即共性下统一的变化和转化规律及关系描述）的科学。精确、简洁、普适是数学研究的最大特点，也是令一代代数学研究者和爱好者最为痴迷和神往的地方。

历史上有关数学方面的专著很多。《几何原本》是古希腊数学家欧几里得的一部数学著作，也是世界上最早的数学方面的专著之一。这部著作是欧几里得几何的基础，在西方是仅次于《圣经》的流传最广的书籍。《几何原本》原著包括 15 卷，涵盖几何、代数、圆、角、多边形、比例、数论等初等数学中最重要的概念。我国最早是在明朝时期由徐光启[22]和利玛窦[23]完成了前 6 卷的翻译工作（如图 3–3 所示），而后 9 卷直到 200 多年后才由李善兰[24]（如图 3–4 所示）翻译完成。

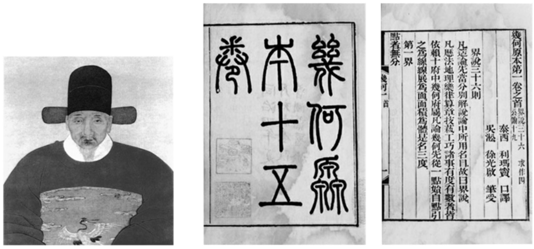


图 3–3　徐光启与《几何原本》中文译本

图 3–4　李善兰

尤其是在中学和大学的数学课上，数学通常给人一种“高傲”且“冷峻”的感觉。我们觉得它所讲述的内容要么晦涩难懂，要么抽象费解，总之就是感觉数学和我们之间的距离非常远。上课时我们学习数学是因为有升学压力，迫不得已，下课后我们做数学作业也同样是因为如果数学成绩不达标就不能毕业——大多数人对数学都没有亲近感。但是，如果真的这么理解就太过狭隘了，或者说是对数学抱有偏见——我们的老祖宗可是有过一些给人感觉大不相同的数学作品。

22　徐光启（1562 年 4 月 ~ 1633 年 11 月），明代著名科学家、政治家。

23　利玛窦（1552 年 10 月 ~ 1610 年 5 月），意大利天主教耶稣会传教士、学者，明朝万历年间来到中国传教。

24　李善兰（1811 年 1 月 ~ 1882 年 12 月），清代著名数学家。

3.2.2 《九章算术》

中国最早的数学专著是《周髀算经》（原名《周髀》），是“算经十书”[25]之一。“算经十书”囊括了汉朝到唐朝 1000 多年间民间数学研究方面的专业著作，也曾经一度作为隋唐时期国子监算学科的教科书。

《九章算术》（如图 3–5 所示）是中国古代科技工作者点滴积累而成的数学专著，西汉的张苍、耿寿昌曾经对其做过增补和整理，成书最迟在东汉前期的公元 1 世纪左右。《九章算术》是中国历史上非常有名的数学专著，并于隋唐年间传往东亚的朝鲜和日本等国。和我们学习的《数学分析》、《微积分》的内容脉络不同，《九章算术》的内容看上去更“接地气”。翻看其中的内容就会发现，《九章算术》简直就是一本中国古代工农业生产应用题大全。

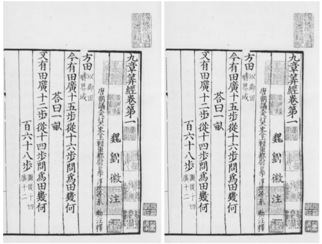


图 3–5 《九章算术》线装本

第一章“方田”——讲的是平面几何图形面积的计算方法，包括长方形、等腰三角形、直角梯形、等腰梯形、圆形、扇形、拱形、圆环等 8 种图形面积的计算方法，还系统讲述了分数的四则运算法则，以及分子、分母、最大公约数等的计算方法。

第二章“粟米”——主要讲述各种谷物的换算比例及算法。最有名的比例算法有 4 项，其中一项为求未知项。该卷列出了求未知项的公式：

$$所求数 = 所有数 \times 所求率 \div 所有率$$

第三章“衰分”——主要讲述比例分配问题。

第四章“少广”——主要讲述包括已知正方形在内的矩形的面积求一边之长等问题，或者已知立方体的表面积求其边长的开方法则。这一章给出的正整数、正分数开平方及开立方的法则，是世界上最早的相关记录。

第五章“商功”——主要讲述以立体问题为主的各种形体体积的计算公式，包括正四棱柱、圆柱、圆台、正圆锥等的体积计算公式。

第六章“均输”——主要讲述以税负和其他应用问题为中心的比较复杂的分配比例计算方法，还提出了有关等差数列的问题。

第七章“盈不足”——主要讲述以盈亏问题为中心的一种双假设算法。

第八章“方程”——这里的“方程”含义与我们现在所讲的不同，专指由线性方程组的系数排列而成的长方阵。除此之外，这一章还在世界上首次提出了负数的概念，说明了正负数及

25 “算经十书”包括《周髀算经》、《九章算术》、《海岛算经》、《张丘建算经》、《夏侯阳算经》、《五经算术》、《辑古算经》、《缀术》、《五曹算经》和《孙子算经》。

零之间的加减运算法则。

第九章“勾股”——主要讲述以测量问题为中心的直角三角形三边互求关系。

《九章算术》最大的特点就是每一个问题的阐述都是一个应用题式的阐述，直接对应当时百姓的农业生产，而且内容通俗易懂，既不故弄玄虚，也不卖弄文墨。我们从里面找两个例子来看看，就能获得非常直观的感受。

1. 挖水渠

例 1[26]

今有穿渠上广一丈八尺，下广三尺六寸，深一丈八尺，袤五万一千八百二十四尺。问积几何？

答曰：一千七万四千五百八十五尺六寸。

秋程人工三百尺，问用徒几何？

答曰：三万三千五百八十二人功。功内少一十四尺四寸。

一千人先到，问当受袤几何？

答曰：一百五十四丈三尺二寸八十一寸之八。

术曰：以一人功尺数，乘先到人数为实。并渠上下广而半只，以深乘之为法。实如法得尺。

译文

今挖渠上宽 1 丈 8 尺，下宽 3 尺 6 寸，深 1 丈 8 尺，纵长 51824 尺。问这段渠的容积是多少？

答：容积是 10074585 立方尺 600 立方寸。

秋季规定每人日工程量为 300 立方尺，问需要多少人？

答：需要 33582 人，其中不足部分为 14 立方尺 4 立方寸。

若 1000 人先开工，问能挖渠多长？

答：能挖渠 154 丈 3 尺 $2\frac{8}{81}$ 寸。

算法

用 1 人工程量的立方尺数乘以先到人数作为被除数；用渠道上下宽度之和除以 2，再乘以深度作为除数。除数除以被除数，即得所挖渠道的长度。

这是一个典型的农业生产灌溉工程中的核算问题。问题的模型是研究一个梯形截面的四棱柱的体积问题。对大多数人来说，直接面对一个光秃秃的四棱柱，可能会没有任何研究的兴趣，而一旦和农业生产相结合，主持工程的人员、当地的村长里正甚至每个参与人员都会关心，因为这涉及生产的时间、效率，进一步当然就是利益问题，所以自然会有亲近感。

26　引自《九章算术》第五章“商功”。

2. 算稻谷

例 2[27]

今有上禾二秉，中禾三秉，下禾四秉，实皆不满斗。上取中，中取下，下取上，各一秉而实满斗。问上、中、下禾实一秉各几何?

答曰：上禾一秉实二十五分斗之九，中禾一秉实二十五分斗之七，下禾一秉实二十五分斗之四。

术曰：如方程，各置所取，以正负术入之。

正负术曰：同名相除，异名相益，正无入负之，负无入正之。其异名相除，同名相益，正无入正之，负无入负之。

译文

今有上禾 2 束，中禾 3 束，下禾 4 束，它们各自之实都不满 1 斗；若于上禾中添加中禾 1 束，中禾中添下禾 1 束，下禾中添加上禾 1 束，则它们分别可得实 1 斗。问上、中、下禾 1 束之实各为多少?

答：上禾 1 束之实为 $\frac{9}{25}$ 斗，中禾 1 束之实为 $\frac{7}{25}$ 斗，下禾 1 束之实为 $\frac{4}{25}$ 斗。

解释

“同名相除……负无入正之”是正负数减法法则。所谓“同名”即同为正数或同为负数；“异名”即两数一正一负；“相除”即相减；“相益”即相加；“负无入”即用负数去减零；“正无入”即用正数去减零。下面用数字表达式来解释一下，设 $a>b>0$。

同名相除：$\pm a-(\pm b)=\pm(a-b)$

同名相益：$\pm a+(\mp b)=\pm(a-b)$

正无入负之：$0-(+a)=-a$

负无入正之：$0-(-a)=+a$

“其异名相除……负无入负之”是正负数加法法则。下面用数字表达式来解释一下，设 $a>b>0$。

异名相除：$\pm a+(\mp b)=\pm(a-b)$

同名相益：$\pm a+(\pm b)=\pm(a+b)$

负无入负之：$0+(+a)=+a$

正无入正之：$0+(-a)=-a$

27 引自《九章算术》第八章“方程”。

算法

依照“方程术”算法，各列置所取禾实之数，用正负算法来运算：（在两行相减时）同号之数则相减，异号之数则相加。正数去减零则变正为负做余数，负数去减零则变负为正做余数。或者（当两项相加时），异号之数相减，同号之数则相加，正数去加零则得此正数，负数去加零则得此负数。

使用三元一次方程组求解。设上、中、下禾之实各为 x、y、z，则依据题设条件列方程

$$2x + y = 1$$

$$3y + z = 1$$

$$4z + x = 1$$

解得

$$x = \frac{9}{25}$$

$$y = \frac{7}{25}$$

$$z = \frac{4}{25}$$

这道题其实多少有一点现代中小学应用题的意味了，它更贴近对模型的提炼，相对不重视实际的应用场景。所以，别看这道题的设置是在核算稻谷的数量，但是在实际生活中不会有人真的这样做。用三元一次方程组求解，我们可以很快得到正确的结果。

在《九章算术》中还有许多这样的例题。例如，在一个虚设的环境中计算代数或几何模型的问题，因为其中绝大部分都基于农业生产场景，所以相对容易在民间流传开来，并对后世产生深远的影响。挖一条水渠究竟需要挖走多少方土，需要多少人，需要多长时间，能引入多少水源——都能直接核算成本及收益。利用“因木望山”的方法（也就是相似三角形的原理）去测量山的高度，也是一种非常实用且简便易行的方法，能有效帮助地理图志的编绘人员降低测量成本（如图 3–6 所示）。如果要缘山建路，也可以通过此法得到参考数据。千万不要小看这些农业数学应用题，它们对我国的农业、手工业和机械制造业都有重要的影响，不仅降低了成本，也提高了效率。

数学本应如此平易近人，而不应是冷峻和高高在上的。让人们世世代代从中汲取营养并茁壮成长，就是数学对人类最直接的贡献。

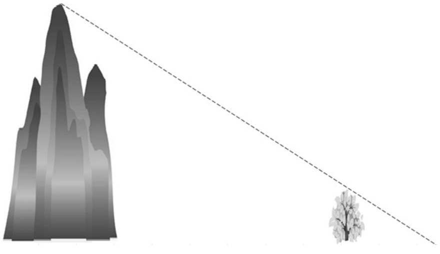


图 3–6　因木望山

3.2.3 高等数学

数学的分支学科非常多，例如代数学、几何学、解析几何、拓扑学、代数拓扑学、统计学、运筹学等，这里还没有列举它们各自的子学科和交叉学科。我们不能不提到数学的一个非常重要的分支——高等数学。通常认为，高等数学是由微积分学、较深入的代数学和几何学及它们的交叉内容所形成的一门基础学科。高等数学的研究范畴包括极限、微积分、空间解析几何与向量代数、级数、常微分方程等。

高等数学研究的是“变量”的数学，而初等数学是指我们在中小学阶段学习的四则运算、平面几何、立体几何、函数、不等式等相对“静止”和直观的数学概念。例如，在初等数学中，我们能用公式计算三角形、四边形、圆形的面积。这些计算公式通常是用“割补法”将它们转化成标准的矩形，然后通过长乘以宽的方式进行计算（如图 3–7 所示）。

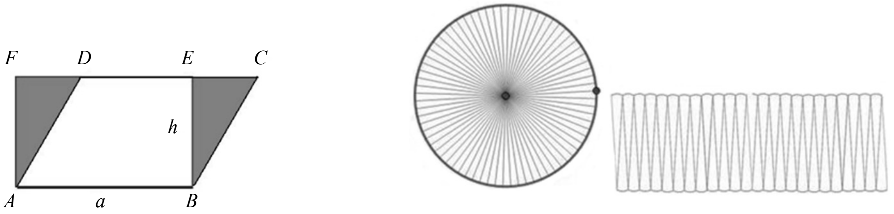


图 3–7　平行四边形和圆形面积割补

求解圆形面积的时候，这种曲线到直线的转化过程已经有一点极限和微积分的感觉了，但理解问题的方式还是一种转化成矩形面积来计算长宽乘积的方式，而用这种方式求解更为复杂的不规则形状的面积是非常困难，甚至是不可解的。

高等数学研究的范畴就是变化中的表达式所通用的解析解获取体系。

举个最简单的例子：如图 3–8 所示，直角坐标系中的两条曲线 $f(x)=\cos(x)$ 和 $g(x)=\sin(x)$，与直线 $x=0$ 围成的不规则图形在 $[0,\frac{\pi}{4}]$ 之间的面积是多少？这个问题在初等数学里是没有解决方案的。但是在高等数学中，应用极限的思想，我们就能比较容易地获得精确的解析解。

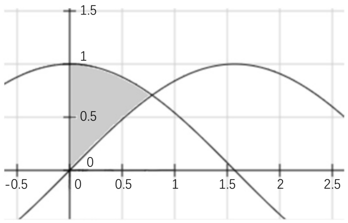


图 3–8　$f(x)=\cos(x)$ 和 $g(x)=\sin(x)$

$$S=\int_0^{\frac{\pi}{4}} f(x)-g(x)$$

$$S=\int_0^{\frac{\pi}{4}} f(x)-\int_0^{\frac{\pi}{4}} g(x)$$

$$S=\int_0^{\frac{\pi}{4}}\cos(x)-\int_0^{\frac{\pi}{4}}\sin(x)$$

$$S=\sin(\frac{\pi}{4})-\sin(0)-(-\cos(\frac{\pi}{4})-(-\cos(0)))$$

$$S=\frac{\sqrt{2}}{2}-0-(-\frac{\sqrt{2}}{2}-(-1))$$

$$S=\sqrt{2}-1$$

从理论上讲，在任意高维的空间中都可以进行这种变化值在极限思想的情况下转换为其他表达式的解析解的方式。那么，任意多维度空间不规则曲面的面积、任意多维度空间不规则体的体积也都可以有精确的解析解了。这种方式为研究现代工业中很多随时间和空间变化的值之间的关系带来了意想不到的收获。

例如，要想精确地计算炮弹的落点，就必须研究发射距离和装药量、爆炸速度、膛压、炮管仰角、风阻、重力等多个因素之间的关系（如图 3–9 所示），而这些因素在整个过程中都是变化的，因此需要建立完整的微分方程组来求解。求解的结果是，这种武器在制式化以后，炮兵可以根据当时的风速，通过调整仰角来计算炮弹的精确射程，提高命中率。不过，这么复杂的问题通常需要使用微分方程组才能解决。

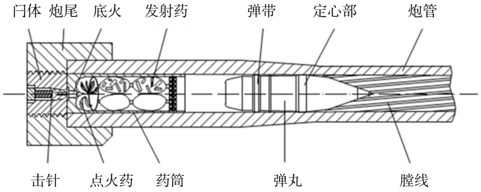


图 3–9　炮弹在炮管中的发射瞬间

连续方程：$\frac{\partial\rho}{\partial t}+u\frac{\partial\rho}{\partial x}+\rho\frac{\partial u}{\partial x}=0$

动量方程：$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+\frac{1}{\rho}\frac{\partial p}{\partial x}=0$

能量方程：$\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial x}+\frac{p(k-1)}{R\rho\psi}\frac{\partial u}{\partial x}=\frac{1}{\psi}(T_1-T)\frac{d\psi}{dt}$

状态方程：$p(\frac{1}{\rho}-\alpha\psi-\frac{1-\psi}{\rho_p})=RT\psi$

边界条件：$u(0,t)=0$

$u(L,t)=0$

$\frac{du(L,t)}{dt}=\frac{dv}{dt}=\frac{Sp_b}{\psi_1 m}$

其中，ρ 代表燃气密度，t 代表时间，p 代表压力，u 代表气体流速，T 代表气体温度，R

代表气体常数颗粒间应力，ψ 代表火药相对已燃体积，ρ_p 代表固相分密度，T_1 代表定容燃烧温度，α 代表余容，L 代表从膛底算起的炮弹位置，v 代表炮弹的平均移动速度，p_b 代表弹底压力，m 代表炮弹质量。

以上是膛内流场的一维均相流方程组，是数学家拉格朗日提出的将复杂的膛内流场简化为一个等截面身管中火药瞬时燃完的纯气体流动问题，并且假设气体密度为均匀分布，在此基础上获得气体动力数学模型的近似解。现代内弹道学研究建模比这个方程组复杂得多，当然计算结果也更为精确。

平心而论，这种计算的技术技巧我国掌握得相对比较晚，在清朝末年和外国列强战斗的过程中，中国军民在列强的舰船利炮面前吃了大亏。高射速的火炮我们造不出来，炮弹我们也造不出来，甚至不知道应该造成什么样子，更别提耐用的蒸汽机或者内燃机了——在复杂计算科学领域的落后带给我们的伤痛真是刻骨铭心。

除此之外，内燃机制造、航空、航天、通信等众多高精尖领域也需要这种计算科学的辅助才能获得成功。例如，在数字通信领域经常使用的傅里叶变换和反变换过程也是高等数学研究的对象，不过它所基于的数学原理更为抽象。最早的电话使用的模拟信号原理是（如图 3-10 所示）：声音通过金属振动膜感应声波来影响磁场和电流，并把这种带有金属振动膜振动“信息”的电流传送到另一端；另一端则进行反向的工作，把不断变化的电流转化为电线圈中磁场的变化，使金属振动膜发出同样的振动。这种信号有比较严重的局限性，一旦距离较远，则电流的衰减会大大影响通话质量。此外，这种方式对信道的浪费也比较大。傅里叶变换和反变换过程解决了音频时域信号到频域信号的转化问题，使数字信号通信代替模拟信号通信成为可能（如图 3-11 所示）。

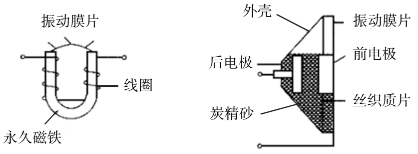


图 3-10　模拟电话听筒与话筒工作原理

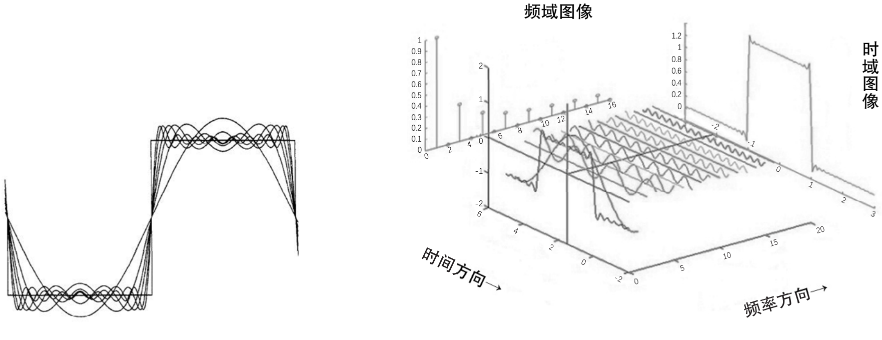


图 3-11　将余弦波叠加成方波

任意一个时域信号 $f(t)$ 经过傅里叶变换，都可以转化成多个余弦波叠加的形式。

$$F(\omega)=\int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

一旦这个环节被我们掌握，语音信号就可以通过调制，由高频载波用极高的速率发送电信号和光信号。

在接收端，用滤波器过滤余弦波信号后，使用傅里叶反变换进行时域信号的还原。

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega t}d\omega$$

这也使信道的安全性、抗干扰性、高速性等得到了基本的保障。信道中不仅可以传输语音信号，一切编码到时域 t 上的数字信号也都可以在高速的信道中传输。我们今天能够使用互联网看网页、听音乐、看视频、和千里之外的朋友聊天，都是拜傅里叶变换和反变换的理论基础所赐。

傅里叶变换和反变换所研究的内容，实际上已经不是数到数的映射关系，而是空间中函数到数的映射关系。这类问题有专门的细分学科进行系统的研究，叫作“泛函分析”。这些抽样领域的深入研究，在与现代化工业建设领域结合的过程中不断产生着新的成果。

现代中国在这些科技领域的发展与进步，很多都得益并依赖于这种建模能力和计算能力的提升，即使没有哪位大师能够再提出类似微积分或傅里叶变换这种有着重大且深远意义的数学思维方式，但只要能够对这些问题有深刻的认识，同样可以在生产中发挥巨大的作用。数学尖端领域的发展是彰显一个国家科技水平的标志，也是支撑现代化建设和高精尖学科的重要支柱，我们对这一点的认识正伴随着科技的发展而越来越真切。

3.3　本章小结

大量低成本的数据在较大的范围内流通，以及平民数学的跨国、跨世纪传承，都在过去的几千年里让人类社会在农业、手工业、纺织业、冶炼工业、化工工业、制造业、信息产业等领域不断获益并产生惊人的“加速度”。而数据与数学的结合，尤其是与先进的计算科学的结合，加上现代制造业为信息产业提供的助力，使数据科学逐渐成为一个相对独立却又与其他行业息息相关的新的科学领域。对这个领域的研究，我们还有很多工作要做，道路还很漫长。

第 4 章　数据科学的使命

这一章我们将讨论数据科学的使命。数据科学是一个巨大而抽象的概念，要想清晰地认识这样一个概念，就需要对其进行拆解。从哪里开始认识数据科学呢？就从它所依赖的最基本的载体——数据——开始吧。

4.1　走近数据科学

数据科学伴随信息技术的逐步发展渐渐羽翼丰满。数据从萌芽到成为一门完整的科学，经历了漫长的过程，形成了今天较为发达的数据科学。然而，我们也不能说数据科学已经非常完备、没有进步的余地了。下面先来回顾一下数据科学的发展历程。

4.1.1　介质

在低成本、高可靠的存储介质形成之前，人类存储数据的方式在我们今天看来是非常落后且效能低下的。以汉字为例，我国最早的文字记载（也就是信息记载）是公元前 14 世纪到公元前 11 世纪左右的殷商甲骨文（如图 4–1 和图 4–2 所示）。

图 4–1　殷商甲骨文

图 4–2　甲骨文和宋体字对照

截至 2012 年，在已经出土的 10 余万片有字甲骨中，大约有 4500 个不同的图形文字，其中已经识别出来的约有 2500 个。这些文字对研究当时人类的主要活动有非常重要的参考价值。

2013 年 7 月，在浙江省平湖市庄桥坟遗址出土了良渚原始文字，这些文字是迄今为止我国

发现的最早的原始文字（如图 4–3 所示）。这一发现表明，距今大约 5000 年前，也就是公元前 3000 年左右，良渚先民就开始使用文字了。这些刻画符号将中国文字史的起点向前推了 1000 多年。

图 4–3　良渚文字

目前发现的这些文字数量非常有限，释义难度也比较大，但仍然可以看出，这些文字应该不是肆意的涂鸦，而是真正具有指代意义的象形文字。世人都期待着有更多佐证性的文字资料能够被发掘，让良渚文字中蕴含的意义为我们所知。

我们的汉字，从甲骨文到金文，再到大篆、小篆、隶书、楷书，一路走到今天，同时伴随着古体字、繁体字到简体字的逐步简化。

我们来看看“得”、“高”、“心”、“虫”4 个汉字从甲骨文到繁体隶书的演化过程（如图 4–4 所示）。结合现在使用的简体汉字的字形，不难看出，字体演化的方向就是从烦琐到简洁，从象形化到符号化，这意味着文字的普及、记录和誊抄的门槛及成本都大大降低，会有更多的人有机会学会汉字，进而使汉字的流通性增强，流通成本降低。文字普及需要一个漫长的过程，但是它和电力、自来水、互联网的普及一样，具有激发民族巨大生产潜能和文化潜能的深远意义。这种演化在其他民族和其他文明的发展中也出现过，并衍生出了东西方几千年交相辉映的文明史——别忘了，这才是数据科学的基础。

图 4–4　甲骨文、金文、大篆、小篆、隶书对照

文字形态不断变化，承载信息的介质也在不断变化。

中国古代的夏朝是一个“传说”中的朝代，很多历史学家，尤其是国外的历史学家，在谈到夏朝时都多有质疑。因为夏朝是一个只在后代历史中提及的朝代，目前还没有带夏朝铭文的文物出土（例如碑文、王侯墓志）——哪怕是一件标有夏朝王侯将相铭文的礼器。目前，考古学家只

是在一些考古遗址（例如“陶寺遗址”[28]）中发现了零星的文物。虽然经碳–14 测年法[29]测定，其年代约为公元前 2500 年 ~ 公元前 1900 年，与历史记载中夏朝的时间（约公元前 21 世纪 ~ 约公元前 16 世纪）有部分吻合，而且遗址中的城址、王墓、观象台、宫殿、仓储区及官方管理的手工业区等遗迹都在一定程度上支持了此地可能是帝尧之都[30]的猜测，但就是由于没有文字方面的佐证，使这种猜测迟迟无法盖棺定论。加之信息存储介质保存技术方面的问题，夏朝没有存世的文字资料，也给解读夏朝的文字和历史带来了极大的困难。

据推测，夏朝留存的文字资料介质可能多为木质、竹质、丝麻质等，在当时的技术条件下，这些有机物很难保存到四五千年后的今天。如果真的是这样，那确实非常可惜。数据信息有时确实向我们展示了它残忍的一面，无论我们抱有多么大的希望，无论从侧面推断夏朝存在的可能性如何之高，只要在数据信息层面没有直接的佐证，也无法认定这件事。

再往后，存储介质的发展逐步取得了高可靠性和低成本两方面的优化成果。虽然过程确实非常漫长，但是从竹简、丝帛、纸，到磁盘（磁介质）、固态盘（半导体介质），信息和数据存储的介质仍然向成本越来越低、质量越来越轻、体积越来越小、速度越来越快、可靠性越来越高的方向发展（如图 4–5 所示）。我们有理由相信，这种趋势还将持续，而且将给数据科学的发展带来越来越多的正面影响。

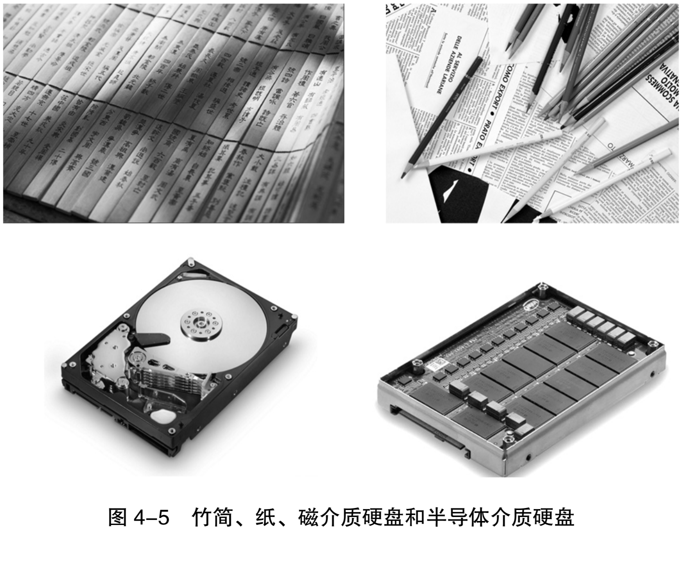


图 4–5　竹简、纸、磁介质硬盘和半导体介质硬盘

28　中国黄河中游地区以龙山文化陶寺类型为主的遗址，还包括庙底沟二期文化和少量战国、汉、金、元时期的遗存。位于山西省襄汾县陶寺村南，东西约 2000 米，南北约 1500 米，面积约 280 万平方米，是中原地区龙山文化遗址中规模最大的一处。

29　根据碳–14 衰变的程度来计算样品大概年代的一种测量方法。这一原理通常用来测定古生物化石的年代。

30　五帝之一尧的都城，建成年代比夏朝还要早。

4.1.2　从信息到数据

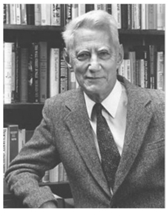


图 4–6　克劳德・艾尔伍德・香农

著名信息学家克劳德・艾尔伍德・香农（Claude Elwood Shannon，1916 年 4 月 ~ 2001 年 2 月，如图 4–6 所示）曾经在 1948 年于《贝尔系统技术期刊》（*Bell System Technical Journal*）上发表论文《通信的数学理论》（*A Mathematical Theory of Communication*），其中有这样的描述：

> If the number of messages in the set is finite then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely. As was pointed out by Hartley the most natural choice is the logarithmic function. Although this definition must be generalized considerably when we consider the influence of the statistics of the message and when we have a continuous range of messages, we will in all cases use an essentially logarithmic measure.

大意就是“信息是用来消除随机不确定性的东西”。香农在这篇论文中不仅给出了信息的含义，还给出了信息量计算的公式，也就是信息学相关专业最为常用的公式之一——信息熵公式。

$$H(x) = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i)$$

其中，$P(x_i)$ 代表消息 x_i 产生的概率。

数据作为介质承载信息的形式，实际上是一种将信息抽象后的符号表示。所以，究其本质，不论是磁带上的语音数据，唱片上的音乐数据，磁盘上的文件数据，还是纸张上的文字数据，都具有如下特性。

（1）作用：承载信息，消除不确定性。

（2）形式：抽象过的符号记录。

（3）定义：符号的含义经过约定，不会或至少不易产生二义性。

信息的流传需要通过存储介质实现持久化。现在我们一提到存储，通常都会想到计算机硬盘，包括传统的机械硬盘及技术越来越成熟的固态硬盘（SSD 硬盘）等。

从世界上第一台多用途电子计算机 ENIAC（Electronic Numerical Integrator And Calculator，电子数字积分计算机，译为“埃尼阿克”，如图 4–7 所示）问世那天起，人类就进入了一个全新的科技领域——计算机领域。

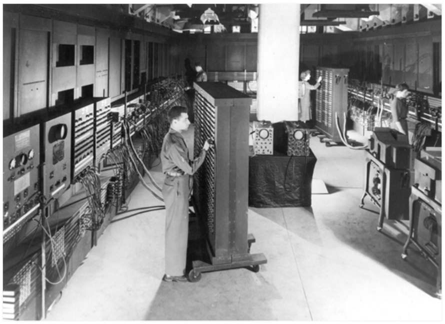


图 4–7　多用途电子计算机的鼻祖——ENIAC

计算机界的“老祖宗”ENIAC，长 30.48 米，宽 6 米，高 2.4 米，占地面积约 170 平方米，有 30 个操作台，重达 30 英吨[31]，耗电量 150 千瓦时，造价 48 万美元（1946 年）。ENIAC 包含 17468 根真空管，7200 根晶体二极管，1500 个中转，70000 个电阻器，10000 个电容器，1500 个继电器，6000 多个开关，每秒可进行 5000 次加法或 400 次乘法运算。其计算效率是继电器计算机的 1000 倍，手工计算的 20 万倍。

第二代计算机出现的时候，生产技术基础发生了改变，人们开始采用晶体管制造电子计算机。国外第二代电子计算机的生存期大约是 1957 年 ~ 1964 年。从第三代计算机开始就是集成电路电子计算机，元器件的尺寸越来越小，集成的规模越来越大。据美国著名计算机零件和 CPU 制造商英特尔（Intel）公布的数字，Ivy Bridge 6 核 i7 CPU 集成的晶体管数量高达 18.6 亿个，工艺规格已经到达 14nm（晶体管和晶体管之间导线的宽度，简称“线宽”）——要知道，一个铁原子的直径也不过 0.25nm。

众所周知，目前的电子计算机使用的计算与存储介质都是二进制的。这不是偶然的结果，而是由电子计算机的实现原理造成的。在 ENIAC 的设计过程中，冯·诺依曼（John von Neumann，1903 年 12 月 ~ 1957 年 2 月，如图 4-8 所示）根据半导体电子元件的二稳态特性，将计算机的计算逻辑设计为二进制方式（即只有“0”和“1”两种状态），在电子管上体现出来的特性就是“截止”和“导通”（如图 4-9 所示）。而人类喜欢使用十进制也主要是因为人有 10 根手指，在日常生活中用十进制进行计算最为便利。

图 4-8 冯·诺依曼

在计算过程中，计算机将“与门”、“或门”、“与非门”、“或非门”等逻辑元件组合在一起，模拟计算进位的过程，进而形成完整的计算实现单元（如图 4-10 所示）。用电子计算机实现的任何一种运算都是由这些逻辑门组合而成的，而这之后在电子计算机领域不断进行的技术改进都是基于同样的原理。尽管元件的体积越来越小，集成的程度越来越高，计算的速度越来越快，但从计算逻辑和实现的数学层面来说，其原理没有本质的变化。

图 4-9 电子管

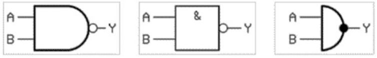


图 4-10 逻辑门

在硬盘技术方面也同样不断进行着量变的积累。无论盘片容量是 500GB 还是 1TB 甚至更高，都是通过磁头机械臂在盘片上滑动的同时高速转动盘片来实现磁性材料持久化和材料状态

31 英国质量单位，也叫“长吨”。1 英吨等于 2240 磅，合 1016.04 千克。

感应（如图 4-11 所示）。计算中使用的“0”和“1”，就可以通过这种持久化以磁介质疏密（磁通量的高低）写入磁性材料层。

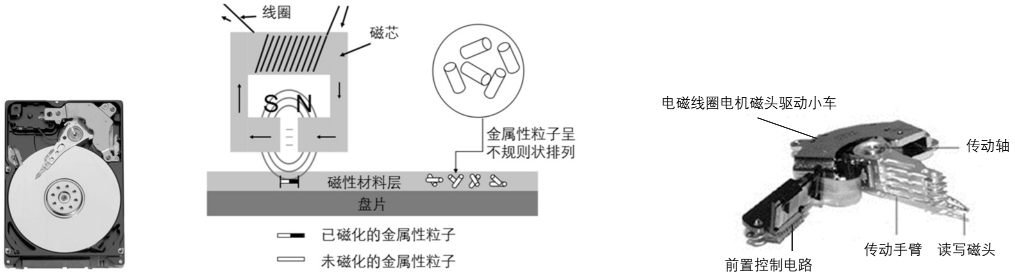


图 4-11　机械硬盘及其读写原理

展望未来，量子计算机（如图 4-12 所示）和光子计算机的研发已经展开。这些新型计算机尽管实现原理不同，但只要成本足够低，而且与目前的电子计算机采用相同的工业标准，使双方在存储介质、信息传输等多个方面能够毫无障碍地互联互通，其未来就非常值得期待。

图 4-12　加拿大 D-WAVE 公司的量子计算机

信息经过抽象、建模，落实到介质上成为数据。数据与不同特性的介质结合，在各种介质上落地的形式也不完全一样，而这种结合通常是数据去迎合介质的特性。有意义的数据落在介质上，才能形成可以传递的信息，这就是信息产业赖以生存的基础。

4.1.3　数据科学的本质

在了解科学和数据的定义后，我们基本可以给数据科学下一个完整的定义了——数据科学是一种研究信息感知、抽象、保存、建模、传输，以及数据之间的逻辑、数量统计、计算和转化关系的综合应用科学。数据科学的本质就是表述和指导对事物认知的关系量化，把普适性的科学思维方式应用到数据上，使其成为一门窄而深的、精确的、拥有完整体系的学科。这就是数据科学要解决的本质问题。

如果尝试对数据科学这个庞大的体系进行细分，可以分为很多专注于数据某一方面处理的

专项学科。例如，数据贮存学研究数据存储中的数据体积、存取效率、可靠性问题等，数据传输学研究传输速度、传输质量问题等。在这两个大的领域中，所有可能影响数据存储和传输的因素都会被讨论和研究，计算机硬件制造、光电信号传输、数据有损/无损压缩、数据冗余与高可用、数据校验等都是研究的对象。

此外，还有研究如何在数据中进行信息抽象、计算、查找、分析等诸多度量与换算问题的数据算法学，以及其他一切能够对数据的感知、抽象、保存、建模、传输，甚至是数据的可视化、数据之间的辩证逻辑提供支持的学科等。其他分类同样会由于产业分化的需要而随时产生。这种平衡中的变化不断发生，一些小的分支和分类也会由于科研工作者潜移默化的努力而逐渐融合为一个大的分支。这些方式同样是仁者见仁，智者见智，不一而足。

4.2 万能的数据科学

伴随着高纯硅提炼技术的进步，以及全球工业化进程的发展，CPU 中二极管的数量越来越多，集成电路的规模越来越大，计算的速度越来越快，电子计算机的制造成本也越来越低。在 IT 界，几乎每个人都知道著名的“摩尔定律”。

摩尔定律是由英特尔（Intel）的创始人之一戈登・摩尔（Gordon Moore，如图 4-13 所示）提出的。他指出：当价格不变时，集成电路上可容纳的元器件的数目，大约每隔 18 ~ 24 个月便会增加 1 倍，性能也将提升 1 倍。尽管这种趋势已经持续了超过半个世纪，但摩尔定律仍应该被认为是一种观测或推测，而不是一个物理或自然法。然而，2010 年国际半导体技术发展路线图表明，这种增长已经在 2013 年年底放缓，之后晶体管数量预计每 3 年翻一番。

图 4-13 戈登・摩尔

关于摩尔定律是否即将失效，目前还无法得出结论。但是，即便摩尔定律失效了，我们也只能说摩尔定律走到了它的边界。在它应验的这几十年中，还是很好地预测了计算能力攀升的速度。不论摩尔定律是否失效，计算机科学的进步都将是一种不可逆的趋势，这能够在很大程度上帮助数据科学应用加速落地。

可以说，数据科学存在于生产和生活的各个方面，贯穿于人类社会发展的始终。正因如此，数据科学也可以称为“万能科学”，因为各行各业的生产问题，以及人类社会中的种种问题，都可以尝试转化成数据科学所擅长的数学问题并通过相应的工具去量化解决。

4.2.1 测量

测量是人类量化认知身边事物的一种手段。人们采用可靠的量具对事物的形状、尺寸、颜色、质量等方面进行度量的行为就是测量。在任何一个年代，任何一个国度，人类都曾尽其所

能采用当时最可靠的量具来进行物体的测量工作。在不同的年代和不同的场合，人类会采用不同的度量手段来保证其可靠性和成本可控。

为了满足不同的测量需求，人类发明了各种各样的测量工具（如图 4-14 所示）。测量长度时，根据精度和单位，我们可以用卷尺来测量，可以用超声来测量，也可以用激光来测量。测量质量时，我们同样可以用不同量程和精度的量具来测量，例如手持式弹簧秤、电子磅秤、地秤等。测量之后，我们把量取的对象及读数记录下来，就基本完成了测量这个工作环节。

图 4-14　毫米尺、电子磅秤、激光测距仪和超声探伤仪

在现代社会，如果我们要对互联网、物联网中的测量对象（分布范围较广、较分散）进行测量，往往需要采用程序检测与报告的方式。

对于可以接入互联网的“节点”来说，只要能够将数据以数据包的方式报送给某个 IP 地址或者域名所对应的服务器就可以了。对使用 Wi-Fi 或者有线以太网的手机或 PC 用户来说，这种条件基本都是天然的了，因为它们所依赖的通信模块和上层的协议栈早已植入终端之中（如图 4-15 所示）。

在道路、旷野或其他有线以太网和 Wi-Fi 无法覆盖的地区也不是没有办法。例如，测量者通常会使用不同规格的 DTU（Data Transfer Unit，如图 4-16 所示）模块进行连接。DTU 向下连接检测设备，例如湿度探测器、温度探测器、PM2.5 探测器、辐射探测器等；向上通过 GPRS、3G-CDMA、4G-CDMA 协议，甚至可以通过卫星信号，以类似拨号上网的方式接入互联网。

图 4-15　有线以太网和无线以太网接入

图 4-16　不同型号的 DTU

这种方式的测量可以说是一种成本最为低廉的方式，几乎不需要任何人工干预就可以坐等各种探测器或者客户端程序把数据收集起来。这种方式基本已经应用于几乎全部的互联网、物联网类产品中，不仅性能稳定，可靠性也能达到了商用水平。

这种测量方式不仅出现在工业领域和一般的商用领域，在民用领域也已落地。现在有很多可以实现远程抄表的智能电表系统已经部署到了数不清的新建小区，也逐步进入老旧小区。还有不少小区安装了具有远程抄表功能的水表和煤气表（如图 4–17 所示）。

图 4–17　具有远程抄表功能的智能电表、水表和煤气表

这些量具同时完成了两件事，一件是测量，另一件是报送。这相当于我们在实验环境中所做的读数和记录工作，只不过读数的动作是由传感器把感应到的测量对象转换为电压、电流等模拟信号，再通过模数转换模块转化成为数字信号的方式实现的，而记录的动作是通过 DTU 模块连接到互联网或者其他专网，然后用数据包报送给服务器的。

目前，最为廉价的实现方式应该还是通过对嵌入式芯片（如图 4–18 所示）进行编程，用 TCP、UDP 或者 HTTP 协议将数据封装成套接字信息[32]或完整的 SOAP 请求[33]传递给服务器。为了保证传输的可靠性，可能会在其中加入一些复杂的控制行为。例如，对数据进行加密，或者针对网络故障导致报送失败时的重传等。

除了刚刚说过的这些常见的测量设备以外，在过去的个半个世纪中，测量酸碱度、PM2.5 浓度甚至咸度、甜度、辣度（如图 4–19 所示）等的仪器都被研制出来，使我们对原本不便量化的领域也有了量化和研究的机会。

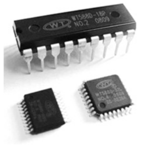


图 4–18　嵌入式芯片

图 4–19　辣度测量仪

32　SOCKET 方式、IP 协议中的一种通信方式。源 IP 地址和目的 IP 地址及源端口号和目的端口号的组合称为套接字，用于标识客户端请求的服务器和服务。

33　简单对象访问协议是一种用于交换数据的协议规范，是一种轻量的、简单的、基于 XML（标准通用标记语言下的一个子集）的协议，它被设计成在 Web 上交换结构化和固化的信息。

应该说，没有量化，就没有科学，更没有数据科学。因此，这种一切都能量化、都要量化的思路是“科学”的基础，是现代各领域研究中非常提倡也不得不重视的根本问题。

4.2.2　统计计算

1. 人口普查

统计计算是一种最为朴素、客观的认知方式。这种认知方式其实是人类潜意识的产物，其出现时间远远早于统计学作为一门完整的学科出现在人类历史舞台的时间。

中国古代的帝王为了了解国家的状况和巩固自己的统治，很早就注重人口普查，这就是最早、最朴素的统计。我猜想，在文字记载比较丰富的周朝，或许周天子就曾问过大臣类似“我大周子民几何”的问题。了解一个国家的人口数量、人口分布、人口结构，就更能有的放矢，对制定国策、颁布法律、鼓励生产、发展军事与手工业等都有极大的好处。

然而，这样的愿望却没那么容易实现。在秦始皇统一中国以前，中国一直长期处于分封制下的邦联式国家形态，虽然中原有周天子作为名义上的“天下之主”，但是周天子的威仪随着时间的推移也慢慢不复存在。据《墨子・告子》记载：“诸侯朝天子曰述职。一不朝，则贬其爵；再不朝，则削其地；三不朝，则六师移之。”这就是西周时期天子分封制中惩戒措施的主要内容：一次不来朝见就贬谪爵位，两次不来朝见就褫夺封地，三次不来朝见就派王师攻打。这是西周天子立下的规矩，听起来相当严苛，只是自从周幽王烽火戏诸侯丢了镐京[34]，东周开始以后，周天子的地盘越来越小，国家经济也越来越差，以至于东周最后一任天子周赧（nǎn）王要攻打自己的臣属秦国还要四处借军饷，最后钱花光了也没能力偿还，当债主们纷纷上门讨债的时候只好筑起一座高台，龟缩在上面躲债——这才有了“债台高筑”这个成语。在这之后就是暴秦统治下的 20 多年，然后就是楚汉相争。总之，国家没有一天是太平的，也就更没法指望在这种环境下进行人口统计了。

历史典籍中第一次出现比较完整的人口记录数据是《汉书・地理志》（如图 4–20 所示），其中有这样的记载：“本秦京师为内史，分天下作三十六郡。汉兴，以其郡太大，稍复开置，又立诸侯王国。武帝开广三边。故自高祖增二十六，文、景各六，武帝二十八，昭帝一，讫于孝平，凡郡国一百三，县邑千三百一十四，道三十二，侯国二百四十一。地东西九千三百二里。南北万三千三百六十八里。提封田一万万四千五百一十三万六千四百五顷，其一万万二百五十二万

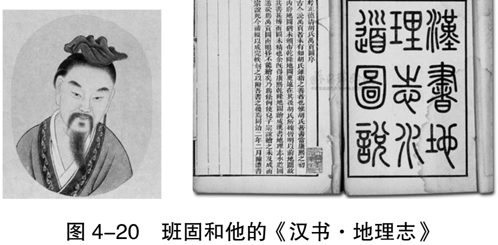


图 4–20　班固和他的《汉书・地理志》

34　中国最早期的城市，现位于陕西省西安市长安区。

八千八百八十九顷，邑居道路，山川林泽，群不可垦，其三千二百二十九万九百四十七顷，可垦不可垦，定垦田八百二十七万五百三十六顷。民户千二百二十三万三千六十二，口五千九百五十九万四千九百七十八。汉极盛矣。”这里对西汉孝平帝刘衎（kàn）（公元前 1 年 ~ 公元 6 年）时期的国土幅员、农田面积、人口情况都做了记录——当时有 12233062 户，59594978 人。

第二次在历史典籍中出现比较完整的人口记录数据是在《后汉书》卷二十九到卷三十三这一部分（如图 4–21 和图 4–22 所示）。

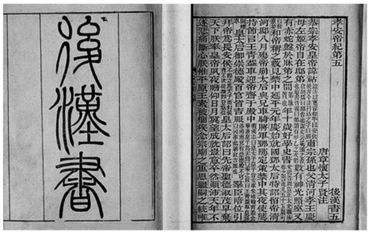


图 4–21　《后汉书》线装本

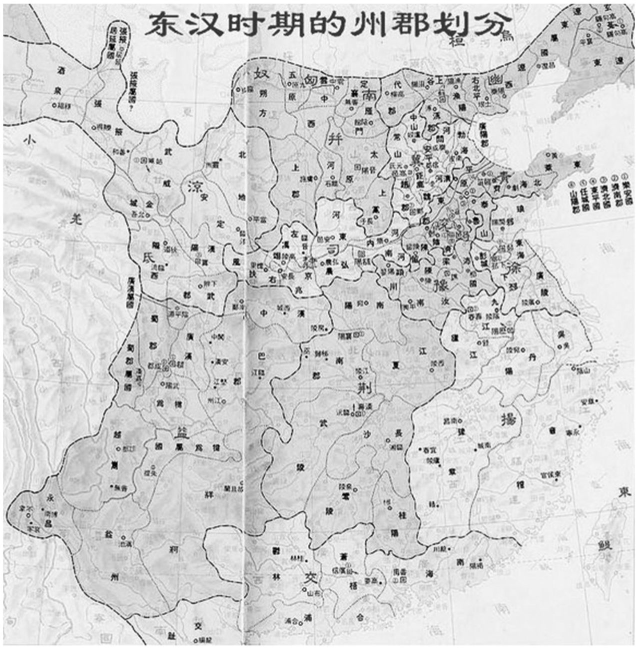


图 4–22　东汉时期的州郡划分

书中记载：“《汉书・地理志》承秦三十六郡，县邑数百，稍后分析[35]，至于孝平，凡郡、国百三，县、邑、道、侯国千五百八十七。世祖中兴，惟官多役烦，乃命并合，省郡、国十，县、邑、道、侯国四百余所。至明帝置郡一，章帝置郡、国二，和帝置三，安帝又命属国别领比郡者六，又所省县渐复分置，至于孝顺，凡郡、国百五，县、邑、道、侯国千一百八十，民户九百六十九万八千六百三十，口四千九百一十五万二百二十。”[36]我们可以知道，东汉孝顺皇帝（刘保，125 年 ~ 144 年）年间，全国共 9698630 户，49150220 人。

书中还有这样的记载：“河南尹秦三川郡，高帝更名。世祖都雒阳，建武十五年改曰河南尹。二十一城，永和五年户二十八万八千四百八十六，口百一万八百二十七。河内郡高帝置。

35　分离、分崩离析之含义。

36　节选自《后汉书》卷三十三志第二十三・郡国五。

雒阳北二十里。十八城，户十五万九千七百七十，口八十万一千万百五十八。河东郡秦置，雒阳西北五百里。二十城，户九万三千五百四十三，口五十七万八百三。弘农郡武帝置。其二县，建武十五年属。雒阳西南四百五十里。九城，户四万六千八百一十五，口十九万九千一百一十三。”[37]

从记述方式来看，这些人口记录不太像是一次国家级的统一人口普查数据，更像是多种不同来源的人口区域性统计数据的汇总。这种统计在没有电子计算机进行辅助的古代，时效性和准确性应该不会太理想，尤其是户数和人数都能精确到个位，这种统计在低位上的可信性其实是值得怀疑的。即使是一次统一性的人口普查，也不会是一瞬间完成的，一定需要一段时间，由基层官吏入户进行统计，然后逐级报送，最后汇总到中央的相关机构——恐怕至少需要数月时间。在这个过程中，人口的自然增减与地域间流动都会对数字产生影响，所以，如此精确的记录反而多少有画蛇添足之感。在这里，前几位有效数字权当参考，当时的人口数大约是 4900 ~ 5000 万。

在这之后，隋、唐、宋、元、明、清都进行过全国性的大规模人口普查，其中明、清两代各有 4 次普查，可见帝王们对国家人口统计的重视程度。

这种统计方法比较朴素，就是纯粹的人口数量汇总，和近现代在计算机技术辅助下进行的人口数量、性别、出生年月、血型、学历、职业等丰富维度的统计完全不同，更不可能用平均值或高斯分布去描述年龄的离散状况。不过，分郡分县进行统计在描述人口疏密程度上其实有比较高的借鉴价值。在地域这个维度上对人口数据做了明显的区分，即使在当时无法通过对统计数据的挖掘来更深刻地认识人口数据，无法进行地域维度的下钻操作，但只要能够合理利用，仍然可以对税收、生产、征兵等工作的科学性给出良性引导。

2. 日志统计

在当今这个现代化的工业与互联网社会，统计再也不是几千年前那种繁重却非常不精确的行为了。前面我们已经了解到，在数据收集和测量的过程中，有大量的工作可以在完全自动化的情况下完成，这本身就大大降低了数据收集和统计的成本。与此同时，减少不必要的人为干预也提高了数据的准确性。

日志统计是目前最为普遍的一种统计行为。这里说的这种日志，不是狭义的类似航海日志一样的专业事件记录，也不是类似计算机服务器里的 Web 容器访问日志（如图 4–23 所示），而是一种广义的日志信息——对客观发生的事情进行描述的记录标志，每个记录标志里可能包括但不限于事情发生的时间、地点、涉及的事物，以及所有涉及的事务对象性状的多维度信息记录。所以，从这个角度来看，超市购物小票是日志，银行卡刷卡记录是日志，手机通话记录是

37　节选自《后汉书》卷二十九志第十九・郡国一。

日志，点菜时的菜品记录是日志，医院里婴儿的出生事件记录是日志……不胜枚举（如图 4-24 所示）。日志的形式和内容是非常宽泛的，在对其进行统计之前，我们需要做好两件事。

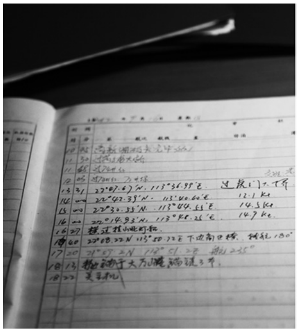


图 4-23　舰船上的航海日志和 Nginx 服务器上的访问日志

胎兒成長就診記錄表

日期	醫院	週數/月	長度(頭臀徑)	胎兒EFW	自費檢查/備註	超音波	體重
98.10.07	有生$200	5w	胚胎 0.85cm		胚胎	1張	47.5
98.10.20	有生$200	6w4d	胎兒 0.72cm		有心跳・拿孕婦手冊	2張	48
98.11.03	有生$100	8w4d	2.00cm(健保給付第一次產檢)		自費$2000 脊髓性肌肉萎縮症篩檢	2張	48
98.11.13	有生$150	10w1d	3.29cm	11/12 凌晨2點多出血・寶寶周圍出現一個血塊		2張	48
98.11.13	天心$100	中醫$1820 安胎藥一星期份(水藥)					
98.11.17	有生$100	11(1)	4.32cm		血塊有長大一點	3張	48
98.11.19	馬偕$530	11			血塊觀察	A4一張	48
98.12.01	有生$100	13w0d	6.79cm		血塊又長大些了	5張	49
98.12.22	有生$100	16w1d	10.34(健保給付第二次)	血塊持續長大(可能是未成形的胚胎)		9張	50
99.01.02	柯滄銘婦產科 16w6d(羊膜穿刺) $7850					2張	
			BPD(寶寶頭蓋的前後徑平均值)				
99.01.19	有生$100	19w4d	4.94cm	490g		9張	52.5
99.02.02	有生$150	21w4d	5.75cm	673g		3張	53
99.02.18	有生$100	24w-1d	(健保給付第三次)			9張	55
99.03.07	有生$0	26w2d			宮縮假日回診安胎無收費	1張	
99.03.09	有生$150	26w4d			羊水過多易早產	3張	57
99.03.17	有生$150			1484g	宮縮安胎	3張	
99.03.18	有生$100	28w	7.61cm(健保給付第四次)			7張	58
99.03.18	有生$200 自費做妊娠糖尿病篩檢						
99.03.19	新光急診$450　99.03.20 中午出院$288　規則性宮縮凌晨辦住院安胎						
99.04.01	有生$100	30w-2d	(健保給付第五次)	1636g	開始2個禮拜回診一次	9張	57.5
99.04.15	有生$100	32w-2d	(健保給付第六次)	1904g	臍帶繞頸	9張	60
99.04.29	有生$100	34w-3d	8.63cm(健保給付第七次)	2197g		9張	61
99.05.11	有生$100	35w+2d	8.94cm(健保給付第八次)	2605g	自費$600 做B型鏈球菌篩檢@臍帶無繞頸	2張	62
99.0518	有生$100	36w+2d	9.04cm(健保給付第九次)	2716g		6張	63.5
99.05.20	有生$150				媽媽感冒了	1張	
99.05.25	有生$100	37w+2d	(健保給付第十次)	2890g		3張	63
掛號費共花費：3818				自費項目共花費：12470		共花費：16288	

图 4-24　购物记录和婴儿出生记录

第一，把这些日志记录以计算机可以识别的方式存储起来。例如，我们通过人工录入、扫描、OCR 识别等多种方式，把原本的纸质记录信息、口述的信息等存储到磁盘上。

第二，把已经转化为数字记录的信息转换成计算机可以直接计算的独立的字段信息。通常在这个环节我们会把数据录入数据库或数据仓库中。

当然，也可以合二为一，一步到位，把信息转入数据库或数据仓库，这个环节称为 ETL（Extract-Transform-Load，也指数据库之间的数据变形与传递工作等）。那么，下一步就是利用数据库或者数据仓库为使用者提供的各种 API 进行统计计算了。

与数据库通常用来做 OLTP（On-line Transaction Processing）工作不同，专业的数据仓库通常用来做 OLAP（On-line Analytical Processing）工作。所谓 OLAP 就是我们平时所说的“数据分析处理”，一般侧重于多维度分析计算，对大量数据处理及数据挖掘操作都有特殊的场景优化。比较专业的数据仓库产品有 Teradata、Greenplum、Hive、Oracle 等，它们对外都提供了非常专业的 API 和分析请求语言接口。

工具的好处就是能抽象出不同应用场景的共同需求，这些数据仓库也不例外。数据仓库除了能在各个产业或业务的应用场景中承载数据收纳与计算的功能，大多配备了专业的数据挖掘组件功能，或者包括建模、分析、可视化功能在内的行业解决方案模板，这样就让各个行业的使用者在应用的过程中通过尽可能少的工作量来完成大部分的分析应用（如图 4–25 所示）。

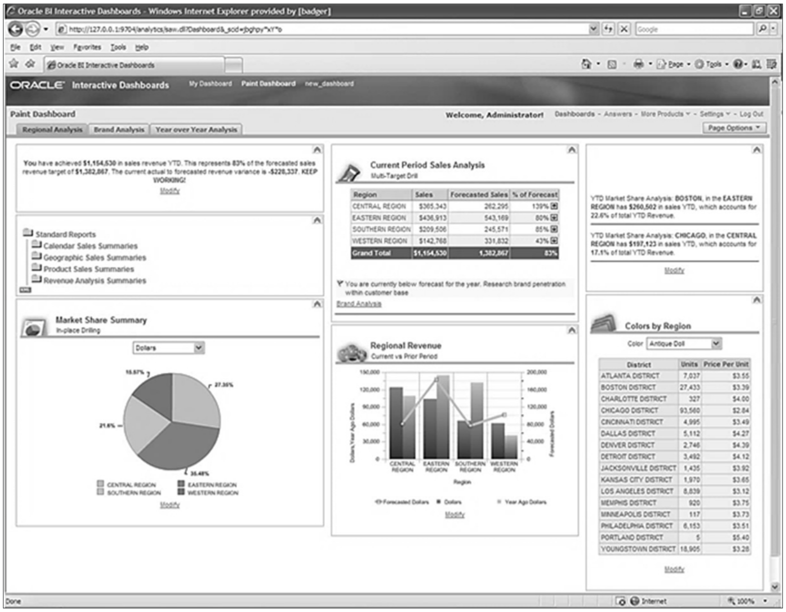


图 4–25　Oracle BIEE Dashboard

无论怎么变化，这些工作无非是在做各种“日志”的统计工作，包括求和、求平均值、求最大值、求最小值、求方差、求比例，求一天、一个月、一个季度、一年的统计值，以及多维度同比和环比。一般来说，无论是对企业，还是对医院、学校、机关单位这样的机构，这些功能已经能够完成日常统计需求中 80% 以上的工作了。对于这些日志在更深层面所蕴含的丰富信息，以及它们彼此之间的关系，可以继续通过一系列数据挖掘算法找到——即使将这个环节算作广义的统计也未尝不可。怎么称呼这个技术环节并不重要，重要的是我们从中获取了哪些有价值的信息。

4.2.3　指标

指标是人们在生产生活中极为常见的一个数据类别。尤其是在工作中，我们会经常见到各种各样用于度量工作效率及质量的指标，例如人们常说的 KPI（Key Performance Indicator，关键绩效指标）。

一般来说，指标是一系列数据经过综合计算的统计值。为什么要使用统计值？统计值的好处显而易见——当大量纷繁的数字出现在阅读者面前的时候，阅读者会非常容易失去焦点，而且阅读大量的数据也需要消耗极高的人工成本。人们在千百年的生产实践中已经潜移默化地养成了使用指标或者统计值来描述宏观事物的习惯，这几乎是一种自然发展的结果。

在前面人口普查的例子中我们已经知道，西汉时期人们就开始对人口数量进行普查和统计，并把这些数字汇总。这就是一种指标——人口数量。这种单一的数字描述代替了对个体的列举描述，不仅简洁，而且抽象出了统计人员最为关心的维度——数量。

世界卫生组织（WHO）一直以来都致力于关心与提高人类的健康水平。那么，全球几十亿人口，究竟应该如何评价健康水平的状况与进步与否呢？平均寿命是一个非常重要的指标。营养健康问题及恶性疾病——这些是在和平时期对人类寿命影响最重大的因素。世界卫生组织的日常工作就是指导和协调国际卫生工作，提供技术援助，消灭流行病，促进并指导生物医学研究工作，以及制定和发展食品卫生、生物制品、药品的国际标准等。

这些工作的成效怎么衡量呢？怎样评价这些工作的成果和进展呢？观察人口平均寿命这个指标就能解决大多数问题。而且，对于有针对性服务或研究的国家，这个指标关注的就不是全球的人口平均寿命了，而是这个国家的人口平均寿命。如果在长期工作中人口平均寿命得到了逐步且明显的提高，则说明工作成效较好；反之，如果在长期工作中人口平均寿命提高的幅度不理想，则说明工作的方法可能存在问题，这时就可以用一些辅助指标来协助判断，查找原因。对全国人口平均寿命这个指标进行分解，也能发现分布规律并找到一些原因。

例如，全国人口平均寿命从性别上可以分为“男性平均寿命”和“女性平均寿命”的组合。

$$\text{全国人口平均寿命} = \text{男性平均寿命} \times \text{男性人口比例} + \text{女性平均寿命} \times \text{女性人口比例}$$

全国人口寿命在“男性平均寿命”和“女性平均寿命”上的体现很可能是不一样的。请注意，在数据分析中，分析人员通常会对这种“不一样”的情形更感兴趣，因为这种“不一样”更有可能引导我们找出导致结果的真正原因。

我们的校园生活中也有这样的事情。不论大考小考，尤其是像模拟考试这种重磅的考试结束后，老师都要做试卷分析。如果一个班级多门课的平均分是 80 分，那么最吸引老师眼球的一定不是得 80 分的学生，而是那些得到 90 多分甚至 100 分的学生和得到 60 多分甚至不及格的学生。让得到 90 多分甚至 100 分的学生分享经验，给只得到 60 多分甚至不及格的学生好好补课，

通常被认为是两种最有效的改进方式。

这种从研究聚合指标深入到研究组成它的独立指标的过程，我们称为“下钻”（Drill Down）。下钻分析在商务智能中应用非常广泛，很多的指标都可以根据同比和环比数据对公司运营和工厂生产情况进行监控。例如，在炼油厂会有“吨油冶炼成本”这种用于监控成本的指标，在互联网企业会有“日活人数”（Daily Active User，DAU）这种用于监控活跃用户数量的指标，在铁路交通部门会有“10 亿吨千米事故率”这种用于监控安全状况的指标等。这些指标能在很大程度上帮助运营人员化繁为简，集中精力，快速发现问题。对各行各业、各种对象的信息，只要合理量化，就能建立合理的指标系统并进行日常运营监控，这才是指标系统的意义。

4.3　使命必达

能说出“有问题”是第一个层次——定性；

能说出“问题在哪里”是第二个层次——定位；

能说出“问题的大小或数量”是第三个层次——定量。

古今中外，各行各业，尤其是手工业、工业、制造业，如果要发展，就必须做到低成本迅速扩张。如果要低成本迅速扩张，就必须做到标准化，然后以极快的速度、尽量低的成本对参与人员进行培训，或者以较低的成本用机器代替人的工作。这两种方式都有一个特点，就是要把那些必须由特定的人来完成的特定的高技巧性工作环节，简化为可以复制的且尽可能快速、低成本完成的工作环节。

4.3.1　高效生产

标准化的好处显而易见——输入的内容是标准的，操作的内容是标准的，那么输出的内容就是标准的，这对输出成果的性状与预期一致有极高的保障作用。

以工人使用的机床为例，人们常说的“车钳铣刨”就是以前工人师傅的 4 个主要工种，即车工、钳工、铣工、刨工，他们都有对应的工具。例如，车工一般使用“车床”作为工具，铣工就用“铣床”，刨工就用“刨床”等。这些车床的使用全凭工人师傅手上的功夫。例如，使用车床的车工，主要是用车刀对旋转的工件进行车削加工，所有的尺寸都在车工的脑子里，所有的操作都由车工通过细腻的手法来调整，从而保证车出的零件光滑圆润。一位车工在上岗前通常需要经过大量的培训及至少几个月的实习，而在实际工作时，精神状态、情绪、疲劳程度等会使工作步骤、尺度把握出现偏差，从而产生一定的废品率。因为与生俱来的生理和心理特性，所以人类很难一直保持稳定的工作状态，不然也不会把这类工作称为“技术工种”了。

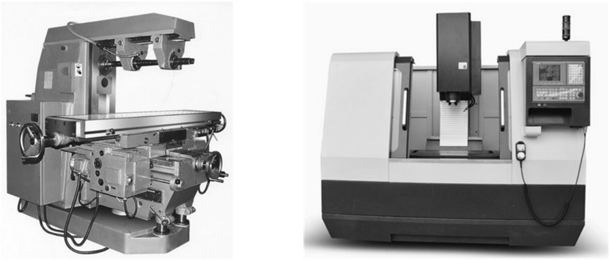


图 4-26　普通铣床和数控铣床

有了数控机床就大不一样了（如图 4-26 所示）。数控机床使用半自动或者全自动的程序进行控制，刀具使用的时间、角度、深浅都像复刻出来的一样，工作 1 次和工作 10000 次得到的产品完全相同。而且，机器不知疲倦，没有情绪影响，在工序复杂的零件加工方面，对效率的提升更加明显。据一些有经验的工人师傅说，对于一些简单的加工工件，使用数控机床能使工作效率提高 3 ~ 5 倍，而对于那些复杂工件，效率可以提升 30 倍甚至更多，且废品率极低。

我们来看一组数据。30 多年来，中国的 CPI（Consumer Price Index，居民消费价格指数）不断攀升，明显地反映了物价上涨的趋势（如表 4-1 和图 4-27 所示）。但是，我们每个人都能感觉到工业品价格的涨幅远没有这么明显，有些甚至在下降。

表 4-1　1979 ~ 2014 年中国 CPI 指数

年　份	CPI 指数	通 胀 率	CPI 累加指数	年　份	CPI 指数	通 胀 率	CPI 累加指数
1979			100.0	1997	2.80	2.8%	425.3
1980	6.00	6.0%	106.0	1998	−0.80	−0.8%	421.9
1981	2.40	2.4%	108.5	1999	−1.40	−1.4%	416.0
1982	1.90	1.9%	110.6	2000	0.40	0.4%	417.7
1983	1.50	1.5%	112.3	2001	0.70	0.7%	420.6
1984	2.80	2.8%	115.4	2002	−0.80	−0.8%	417.3
1985	9.30	9.3%	126.1	2003	1.20	1.2%	422.3
1986	6.50	6.5%	134.3	2004	3.90	3.9%	438.7
1987	7.30	7.3%	144.1	2005	1.80	1.8%	446.6
1988	18.80	18.8%	171.2	2006	1.50	1.5%	453.3
1989	18.00	18.0%	202.1	2007	4.80	4.8%	475.1
1990	3.10	3.1%	208.3	2008	5.90	5.9%	503.1
1991	3.40	3.4%	215.4	2009	−0.70	−0.7%	499.6
1992	6.40	6.4%	229.2	2010	3.30	3.3%	516.1
1993	14.70	14.7%	262.9	2011	5.40	5.4%	544.0
1994	24.10	24.1%	326.3	2012	2.60	2.6%	558.1
1995	17.10	17.1%	382.0	2013	2.60	2.6%	572.6
1996	8.30	8.3%	413.8	2014	2.00	2.0%	584.1

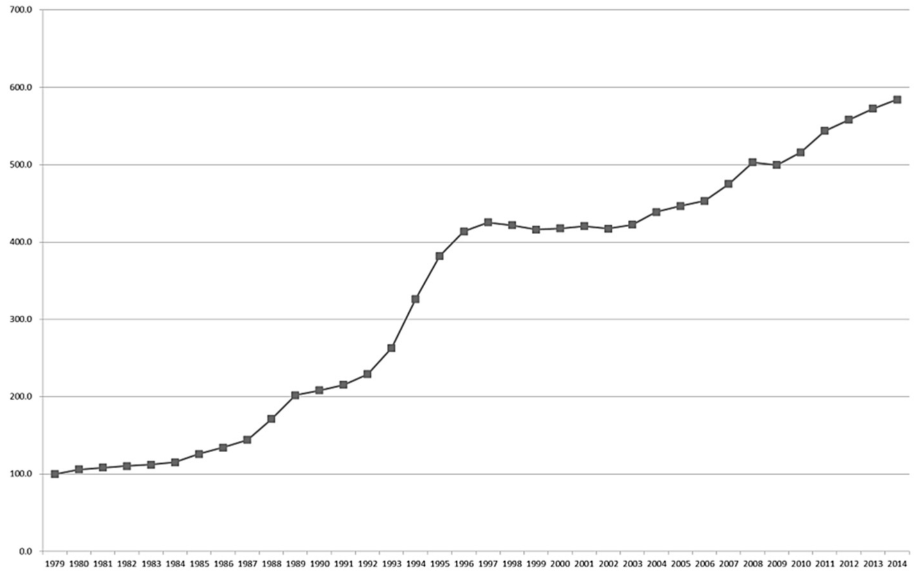


图 4–27　1979 年至 2014 年 CPI 累加指数[38]

表 4–1 展示了 1979 年到 2014 年中国 CPI 指数的统计信息，图 4–27 则把指数累加值转化成了曲线图。从图表上看，1979 年 CPI 累加指数为 100，而 2014 年为 584.1。也就是说，1979 年价值 100 元的物品在 2014 年价值已经达到 584.1 元；或者说，2014 年的 584.1 元在 1979 年只能当成 100 元使用。

就算不看这些数据，我们自己也会有感觉——这些年有些东西的价格上涨得特别快（例如农业产品），但是也有不少东西的价格确实在下跌。1983 年，国内组装桑塔纳汽车的价格约为 30 万元，我们算一算这笔钱在 2012 年能“兑换”成多少元——大约是 149 万元。2012 年，大众新桑塔纳的平均价格约为 10 万元，这笔 149 万元的“巨款”几乎可以买到 15 辆新桑塔纳！这种反差是十分惊人的。

为什么有这么大的差距呢？一个很重要的原因就是工业品生产能力的提高远高于 CPI 的增长速度。工业品为什么能有如此巨大的生产能力提升呢？其中很重要的因素就是生产技术的自动化，以及大规模生产线的使用，而更为重要的因素是标准化。这种标准化建立的基础就是期望用数字对生产条件进行约束，以达到生产条件一致，进而达到生产成果一致的情形。这种模型的稳定性对于资本的吸引力是明显的，也是巨大的——谁都喜欢这种算得明白、预期应验性高的买卖。要讨论为什么稳定性会有如此大的吸引力，就几乎等同于讨论在全社会范围内是劳动谋生靠谱还是买彩票谋生靠谱。我想答案不言自明——这不正是数据科学的成就吗？

38　来自百度文库。

4.3.2 破除迷信

人们热衷于进行各种数据观测、拟合，希望对数据进行可期望的预判。这种行为本质上是在做什么呢？用带有一定功利色彩的眼光来看，这实际上是一个趋利避害的过程。

在我国古代科学技术尚不发达的时候，巫蛊之术盛行。人们祈求神佛保佑，相信自己的虔诚能够感动神灵，赐予自己实现愿望的能力——这就是人们经常说的“保佑（某人或某事）”。直到现在，人们仍然分成两派，分别坚持自己的观点。

观点一：这完全是封建迷信，是愚昧的表现，根本不可信。

观点二：神灵是存在的，如果不信，只是因为“凡夫俗子”没有这种感悟，或者不够虔诚，所以无法打动神灵来保佑。

这两种观点，即使不是极端敌对，非要除对方而后快，也至少是互相不以为然，绝无调和的可能。

在本书最开始的部分，在对科学家的特征进行归纳的过程中其实已经明确过了——科学是测量的学科，是量化计算的学科，而且即使不具备突破性，也需要精确性和体系性这两个最为基本的特性。那么，我们不妨想想，我们信任科学的根本原因是什么？科学能够精确地通过量化地进行“应验”，而这个过程中对数据的观测、拟合，以及对模型的归纳，又是在做什么呢？调整的最终目的是为了让模型所表达的“体系”和观测的现象一致。这个“一致”是非常重要的，因为不“一致”就是不能“应验”。

对于不能“应验”的东西，会有人感兴趣吗？我不知道，因为我也确实看到，从古至今无数人对玄而又玄的东西感兴趣，甚至对“通灵”感兴趣。尤其是在中国魏晋南北朝时期，士大夫阶层都非常热衷于谈论玄学。然而，这种东西在民间和商界不容易传开也是有原因的，恐怕不能单单归咎于除士大夫阶层以外的人都蠢笨之极——也确实不可能是这样。不能“应验”的东西一定不能用于大规模生产的普及，也不能用于生产技术的改进，而不能直接或间接发展生产力的东西通常经不住时间的考验，也不会在整个社会受到广泛认可。不用举太多例子，不论是什么主义，什么理论，什么技术，如果最终不能对发展生产力有帮助，则大都昙花一现，很快就会淹没在历史的洪流之中，或者被束之高阁，成为人们茶余饭后的谈资，并不会得到普及。

再回来看看那些虔诚祈求神灵进行帮助的行为。有人能说得清需要使用多少量的虔诚能够获得多少量的神灵的帮助吗？退一步讲，假设这种关系或者规律客观存在，那么它能被丈量吗？如果无法丈量，就无法得到量化关系。这种行为的危害是什么？因为这是一个无法自圆其说的体系性解释模型，所以后面的事情就没办法指望了。例如，精确地预测神灵的庇佑量，模型的改进也没办法期许——经过怎样的改进能够使神灵的庇佑增加 10% 或更多？这种极为“不稳定”的输出与输入关系使可预测性极差，自然是不能商用的，因为量化的商业模型不存在，也就不知道该投入多少资金，耗费多少时间，获得多少收益。这些不可期许的利益对天生必须靠打算

盘来核算成本和收益的商人来说是绝对无法接受的——没办法，还得回去琢磨怎么从原料和人工上省钱，怎么通过满足客户偏好甚至市场的宏观需求来多赚钱。不仅如此，估计连普通人也不会再热衷于对这些事情进行研究，除非完全当成业余爱好——这就跟科学没有任何关系了。

目前世界范围内没有出现过任何一个能够量化虔诚和庇佑关系的模型，哪怕是萌芽性的观点都没有。所以，这种东西我们没办法认定其为科学，并不是因为它是从远古蒙昧时期流传下来的，而是因为它不符合科学的定义。即使真的有这种关系，即使有相当数量的人笃信神灵与庇佑的存在，只要它不能量化，不能进行量化的验证，就不能被认定为科学。补充一句，严谨地说，我同样不能斩钉截铁地否认它的存在。从科学的角度来讲，至少我无法证明它一定不存在——它，只是不科学。

4.3.3　目标一致与不一致

说到“使用数据科学来求解问题”这个课题，我们必须认同一个重要的观点——出发点必须一致。如果出发点不一致，每个人都站在不同的观点上对事情进行评价，每个参与的人都用自己的价值观和利益观点来进行评价，就无法得到公允的解决方案——因为公允的评价方式不存在。怎么办？还是要回归到简化模型的观点上来——定义目标，量化目标。

在机器学习领域有一个概念叫作“损失函数”，相信对机器学习有过一些研究的读者朋友不会感到陌生。其实，不仅是机器学习领域，在数据分析和工程优化等领域都有这个概念存在。所谓损失函数就是用来衡量和比较多种方案的优劣程度的量化工具。对于一个问题的多个解的比较过程，只要比较损失函数的值就可以了——这是一种极为理想的方案。

损失函数根据定义，会有一个对评价的完整解释，例如“损失函数越小，说明解越好”。在线性回归中的表示方法如下：

$$e=\sum_{i=1}^{n}|y_i-(ax_i+b)|$$

这个模型希望把误差 e 描述成所有的样本值 y 与 ax_i+b 的差，也就是 e 为 0 的时候最为理想，拟合没有误差[39]。当然，误差 e 越大，说明拟合失真越大，拟合效果越不好。

$$Loss=\sum_{i=1}^{n}|y_i-f(x_i)|$$

在机器学习领域，每种模型的训练都需要定义 *Loss* 函数，它表示的意义通常也是拟合结果与期望结果的差值的和——当然也是越小越好，和前面说的 e 的定义是一样的。在不同的场合，这种 *Loss* 函数的形式可能会不同。在损失函数的比较中，会认为其值越小越好；而在其他场合，会定义一些越大越好的目标函数来解释“收益”的问题，这种情况下函数值越大越好，考试成

39　这种情况下，通常要注意是否会出现过拟合。

绩就是一种常见的目标函数。它们所完成的功能都是“评价”，即量化评价一个方案的效果如何。

目标的一致化给问题的讨论和求解带来了可能。这种方式给我们带来的好处在后面的章节还会具体讨论，现在我们只要知道这种目标一致的思路是数据科学解决问题思路的重要元素就可以了。

4.4 本章小结

数据科学首先作为科学，应该满足科学的基本定义。在此之上，数据科学还兼具数据本身的一些特性。

数据作为信息的载体，通过在介质上的落地和传输，以及数据之间体系性的计算，给人类带来了“消除不确定性，降低试错成本”的巨大利益。在本章中，我们讨论的是数据科学的宏观价值与意义。至于整个数据科学在现代生产中如何落地，以及在每个环节应当注意的问题，在“实践篇”还会专门讨论。

第 5 章　矛盾的世界

在我们所存在的这个世界上，充满着矛盾，各种各样的矛盾。

人们所说的矛盾，其实是指事物相互影响和对立的关系。简单地说，我们通常认为：如果有两件事物，其中一件（甲）存在，则另一件（乙）也存在；如果甲不存在了，那么乙也不存在了。如果有这样的一对事物，我们就称甲和乙是一对矛盾，或者说甲和乙之间存在矛盾。

对矛盾的研究是哲学研究的重要方面，从古希腊文明最辉煌的公元前 800 年左右，就涌现出一批又一批的哲学家对这个矛盾的世界不断进行研究。

5.1　古希腊——学者高产的国度

“古希腊”和我们今天所说的“希腊”其实不是一个国家。作为西方文明的主要源头之一，古希腊文明持续了约 650 年（公元前 800 年 ~ 公元前 146 年），是西方文明最重要和最直接的渊源。古希腊地跨爱琴海东西两岸，国土覆盖小亚细亚的西部沿海地区、希腊半岛、爱琴海诸岛等（如图 5-1 所示）。公元前 146 年，古罗马征服了古希腊，从此古希腊灭亡。

图 5-1　古代希腊地图

在古希腊延续的约 650 年，正好是中国西周末年、春秋、战国、秦直到西汉初年，是两个

世界上最重要的文明古国在思想文明领域最为璀璨的时期之一。在这个时期，古希腊诞生了十几位世界著名的数学家和哲学家。

毕达哥拉斯（Pythagoras，约公元前 580 年 ~ 约公元前 500 年，如图 5–2 所示），古希腊数学家、哲学家。他的最大贡献是在公元前 6 世纪提出“地圆说”（也就是“地球是一个球体”）的概念及毕达哥拉斯定理（也就是“勾股定理”）。虽然毕达哥拉斯定理在提出时间上晚于考古中出土的公元前 2000 年古巴比伦人在泥板书上用楔形文字对勾股定理所做的相关记录，但西方世界还是普遍承认毕达哥拉斯对发现勾股定理所做的贡献。

柏拉图（Plato，约公元前 427 年 ~ 公元前 347 年，如图 5–3 所示），古希腊伟大的哲学家。柏拉图虽然在数学方面没有建树，但他是公认的西方最伟大的哲学家和思想家之一。他和老师苏格拉底、学生亚里士多德并称“希腊三贤”。柏拉图一生著述颇丰，他的哲学思想主要集中在《理想国》和《法律篇》两部著作中。

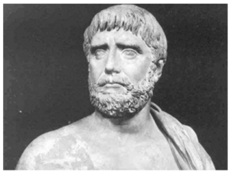


图 5–2　毕达哥拉斯雕像和楔形文字记录的勾股定理

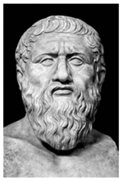


图 5–3　柏拉图雕像

柏拉图的《理想国》以辩论记录的形式成书，在第七卷中用著名的洞穴比喻来解释理念论。

有一群囚犯在一个洞穴中，他们手脚都被捆绑，也无法转身，只能背对着洞口。他们面前有一堵白墙，身后燃着一堆火。在那面白墙上，他们看到了自己和身后到火堆之间事物的影子。由于看不到任何其他东西，所以这群囚犯以为影子就是真实的东西。最后，一个囚犯挣脱了枷锁，摸索着找到了洞口。他第一次看到了真实的事物。他返回洞穴并试图向其他人解释那些影子其实只是虚幻的事物，并为他们指出光明的道路。然而，对还在洞穴里的那些囚犯来说，这个逃出去的人似乎比他逃出去之前更加愚蠢——还在洞穴里的那些囚犯向逃出去的人宣称：除了墙上的影子之外，世界上没有其他东西了。

这种思想在相当程度上鼓励人们不要被当前观察到的现象所限制，也不要感到困惑，要不断反思和求真。这种哲学思想为后世尤其是古希腊后世的思想进步和学术进步带来了巨大的正面影响。

除此之外，欧多克斯（Eudoxus），亚里士多德（Aristotle）、欧几里得（Euclid）、阿基米德（Archimedes）、阿波洛尼乌斯（Appollonius）、托勒密（Plolemy）、刁藩都（Diophantus，也译作“丢番图”）等著名学者也在不断地为世人贡献自己的智慧。

在这里，我们除了列举数学家，还列举了不少哲学家。不要觉得这很奇怪。有这么一句话很流行——物理学的尽头是数学，数学的尽头是哲学，哲学的尽头是神学。这句话的出处现在没法考证了，不过，这同样是仁者见仁，智者见智，或以为对，或以为错，其实不要紧。在我看来，这其实表现了人类认知边界拓展过程中的一种从量变到质变的过程，同时也从一个侧面说明人类认知世界过程中在边界上的迷茫和困惑。

5.2　矛盾无处不在

矛盾的存在早已超过了数据存在的范畴，并在人类生产、生活的方方面面出现。它影响着我们判断事物的思路，也指导着我们前进。

图 5–4　老子和精装本《道德经》

世界也是因为有各种矛盾才变得生机勃勃，变化万千，令人心醉神迷——所谓“阴阳生万物”。中国古代著名的哲学家老子[40]在其传世名作《道德经》[41]（如图 5–4 所示）中写道：“道生一、一生二、二生三、三生万物。万物负阴而抱阳，冲气以为和。”这句话直接翻译成现代汉语恐怕读起来还是很晦涩，而且在释义方面也莫衷一是，各家有各家的见解，我们只要理解它的大意就好。“道生一”中的“道”是指万事万物（包括一切事物，不管是已知的还是未知的）运行的客观运行规律或者秩序。“道生一”是指从混沌洪荒状态发展到出现第一个可辨识的状态，或者我们所认为的维度、物质等。“一生二、二生三、三生万物”是指一种通过这些最简单的物质裂解、组合，产生大千世界的过程。“万物负阴而抱阳，冲气以为和”是指万事万物都是背对“阴”的一侧而面朝“阳”的一侧，阴阳二气交融，产生“和”气并生发出来。

这种阴阳平衡、万物繁荣的理念几乎贯穿在所有的事物当中。原子有正电子和负电子，所以产生了平衡。原子和原子之间由于电子得失产生的价态变化形成了各种物质。由于日照不均匀，导致地球上有的地方气温高，有的地方气温低，而且高低悬殊，所以形成了水的蒸发。水蒸气由风运送到其他地方，由于引力和浮力不平衡而产生降水，才有了陆地上的江河湖泊，以及丰富的陆地生命形态。这大概就是阴阳生万物最浅显的体现方式吧。

数学作为辩证哲学的一种表现形式，其中也蕴含着丰富的矛盾。例如，我们常常说的实数里面就有正数、负数和零，日常生产生活中的数学应用几乎就是正数、负数和零组合形成的变化万千的内容表达，丰富且有趣。正数和负数就可以看成矛盾的两个方面，而且这两个方面还可以相互转化。

40　老子，姓李，名耳，字聃。

41　又名《道德真经》、《老子》、《五千言》、《老子五千文》。

与实数相对的还有虚数。虚数用来补充原来在实数中没有定义的内容。

$$i^2 = -1$$

这种原来在实数范围内没有定义的数字有了定义，此外还有了一系列的相关定义。

$$i^1 = i，i^2 = -1，i^3 = -i，i^4 = 1，i^5 = i，i^6 = -1\cdots\cdots$$

虚数的意义在于它与实数结合后形成复数，进而形成更为丰富的计算含义。例如，$1+3i$ 就是一个复数，它表示一个二维矢量，这个矢量从 (0, 0) 指向 (1, 3)（如图 5–5 所示）。如果用它来表示空间力矢量的话，计算就很方便了。

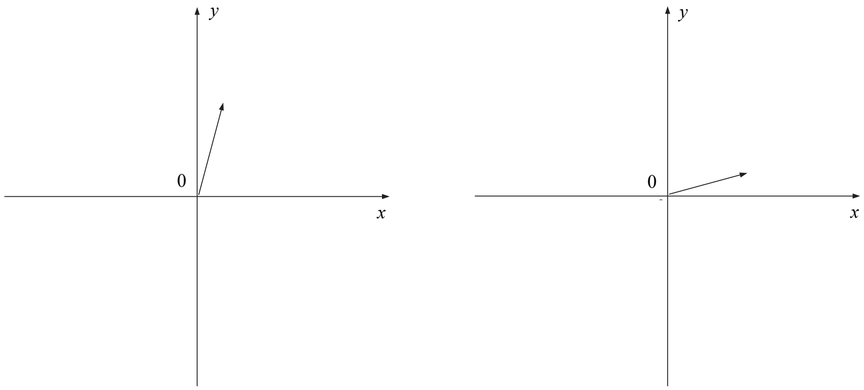


图 5–5　矢量 $(1,3i)$ 和矢量 $(3,i)$

如果空间上还有一个力矢量指向 (1, 3)，那么根据平行四边形法则，两个力的合力应该指向 (4, 4)，就是把两个复数的实数部分和虚数部分分别相加（如图 5–6 所示）。

可以说，这种计算方式非常方便，也是虚数参与运算的一种加法形式。除此之外，还有乘法形式，用来表示旋转角度的问题，使用起来也非常方便。

在数学上，我们定义了加法，又定义了它的矛盾面——减法；我们用加法定义了乘法，又用加法的矛盾面减法定义了乘法的矛盾面——除法；我们用乘法定义了乘方，又用乘法的矛盾面除法定义了乘方的矛盾面——开方；我们用加法和乘法定义了积分，又用减法和除法定义了它的矛盾面——微分。这不就是一个阴阳生万物的过程吗？也正因为有了这一系列的矛盾的计算方法，才有了丰富的近现代计算科学成果。矛盾真是个伟大的存在。

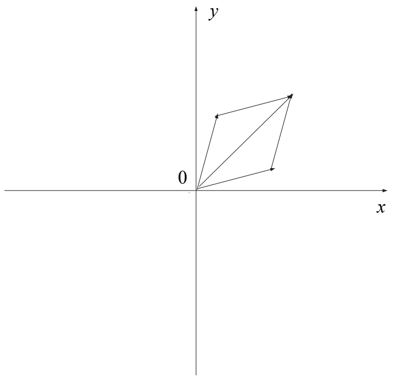


图 5–6　矢量 $(1,3i)$ 和矢量 $(3,i)$ 形成合力 $(4,4i)$

5.3　世界究竟是否可知

世界究竟是否可以为人类所知呢？千百年来，这个观点一直让人无比困惑。诚然，在人类文明发展的漫漫征途中，不断取得各个认知领域的进展与突破——天文学、数学、农业科学、地理学、气象学等。人类对自然、对所存在宇宙的认知也在不断进化。虽说人类在自然科学中的理论推导和进步起着相当的作用，但是理性的科学家对此有着不相同的见解。现代的科学家比古代的科学家更聪明吗？这话真不好说，毕竟几千年来古今中外的科学家们还从来没有机会来个脑力大比武。既然如此，就让我们从历史典籍和各种出土文物中去找一找根据吧。

先说说太阳系的“九大行星”吧——说“八大行星”可能更严谨一些。在 2006 年 8 月 24 日于布拉格举行的第 26 届国际天文联会中通过的第 5 号决议中，冥王星由于性状与行星不符，所以被划分为“矮行星”，从太阳系“九大行星”中除名。从此，太阳系的“九大行星”就变成“八大行星”了，相关的教科书内容应该也做了相应的修改。

太阳和月亮自不必说，一个是距离地球最近的恒星，一个是地球唯一的卫星——距离地球最近的天体，我们想不发现都不行。有据可考，早在公元前 3000 年的苏美尔时代[42]，人们便发现了水星。不是出于什么特别的原因，而是它距离太阳最近，反射出的太阳光极强，我们用肉眼就能轻易看到它。由于金星、火星、木星、土星的体型较大，所以我们通过肉眼及低倍数的望远镜就能观测到（如图 5–7 所示）。

1781 年 3 月，弗里德里希·威廉·赫歇尔（就是我们前面提到的发现光分七色的那位科学家）发现了天王星。他用自制的 15 厘米口径的反射望远镜观测双子座的时候，发现一颗陌生的星，这颗星比较亮，可是在星图上却查不到。在改用 400 倍和 900 倍的目镜仔细观测之后，确定这不是一颗恒星。慎重起见，赫歇尔连续 10 个夜晚密切地关注着这颗星，发现它每天都在缓慢地移动（虽然不容易察觉）。他最初向皇家天文学会报告说新发现了一颗彗星，但彗星一般都沿着比较长的椭圆轨道运行，在离太阳很近的时候还会“长”出“尾巴”。可是，这颗新星却不是这样，它的边缘一直比较清晰，丝毫没有彗尾特征，而且它的轨道似乎接近圆形。不久以后，赫歇尔根据观测数据计算出它的轨道近似圆形，距离太阳比土星远了约 1 倍。这时，他意识到自己发现了一颗新的行星。后来，法国天文学家拉普拉斯算出这颗新星的轨道距太阳 19.18 天文单位[43]。经过一段时间的观测之后，天文学界终于确定它是太阳系的一颗行星，它与太阳之间的距离比当时所知最远的行星——土星的距离远 1 倍。赫歇尔发现新行星的消息引起了极大的轰动。这颗行星被命名为“天王星”（Uranus，乌拉诺斯神，取自希腊神话中土星父神的名字，如图 5–8 所示），赫歇尔也因此一举成名。

42　美索不达米亚文明的一部分。巴比伦时代（公元前 3500 年 ~ 公元前 3200 年）之前是苏美尔时代。

43　1 天文单位为日地平均距离，约等于 1.49 亿千米。

图 5–7　老式反射望远镜

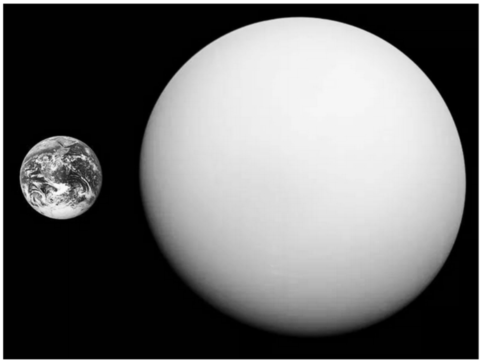


图 5–8　直径 52000 千米的天王星体积约为地球的 64 倍

其实所有的人也都知道，在赫歇尔发现天王星之前，天王星肯定已经在那个位置围绕太阳运行了千百万年了。以前的人没有发现它，不是因为他们都愚蠢且不懂科学，而是受限于科学认知维度的缺乏和观测手段的落后。

后来，当我们拥有巨型射电望远镜之后，就能够观测到 130 亿光年以外的天体了，人们所能识别的天体数量在这个过程中也迅速增加。2003 年，国际天文联合会在悉尼召开的会议上公布，目前已知的恒星数量大概是 700 万亿亿颗，也就是大约 7×10^{23} 颗。

从原始的“天圆地方”、“日月金木水火土”，到如今的 700 万亿亿颗恒星，人类不曾创造，人类的进步仅仅是在感知手段和水平方面。所以，现在也有人说“人类不曾发明任何东西”，言外之意是一切事物早已“客观存在”，只是我们还没有“发现”——这应该是对边界感知能力在人类生存中作用的最夸张的说法了吧。

其实这也说明一个问题——在任何时刻，我们都不能说我们“看到”的一切就是一切，因为我们的感知能力相对这无垠的宇宙是那么有限。

如果我们从数学的角度来认识这个事实，也好有一比。不管在数轴上取一个数值多么大的点，在它的右侧还有无穷多的数字比它的数值大；不管在数轴上取一个数值多么小的点，在它的左侧还有无穷多的数字比它的数值小——我们观测到的数轴上的数值范围，与正负无穷相比是那么有限。这就是人类在不断改进自己认知条件的过程中常常遭遇的困惑，也是人类不断改进自己认知条件的最大动力。

5.4　薛定谔的“喵星人”

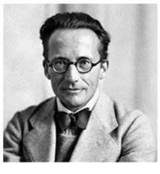


图 5–9　埃尔温・薛定谔

埃尔温・薛定谔（Erwin Schrödinger，1887 年 8 月 ~ 1961 年 1 月，如图 5–9 所示）是量子物理领域杰出的物理学家，奥地利人，曾在瑞士苏黎世大学、德国柏林大学和奥地利格拉茨大学任教授。

薛定谔对量子物理界的最大贡献是以他的名字命名的“薛定谔方程”，也叫“薛定谔波动方程”（Schrodinger Wave Equation）。

一维薛定谔方程：$-\frac{h^2}{2\mu}\frac{\partial^2\psi(x,t)}{\partial x^2}+U(x,t)\psi(x,t)=ih\frac{\partial\psi(x,t)}{\partial t}$

三维薛定谔方程：$-\frac{h^2}{2\mu}(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2})+U(x,y,z)\psi=ih\frac{\partial\psi}{\partial t}$

定态薛定谔方程：$-\frac{h^2}{2\mu}\nabla^2\psi+U\psi=E\psi$

最后一个方程是描述一个粒子在三维势场中的定态薛定谔方程。所谓势场，就是粒子在其中会有势能的场，例如电场就是一个带电粒子的势场。所谓定态，就是假设波函数不随时间变化。其中，E 是粒子本身的能量；$U(x, y, z)$ 是描述势场的函数，假设不随时间变化。薛定谔方程有一个很好的性质，就是时间和空间部分是分立的，求出定态波函数的空间部分，再乘以时间部分 $e^{-\frac{tiE\cdot 2\pi}{h}}$，就形成了完整的波函数。不过，今天我们不研究这么专业的内容，而是讨论一个有趣的话题——薛定谔的猫。

“薛定谔的猫”是由奥地利物理学家薛定谔于 1935 年提出的有关猫生死叠加的著名思想实验，是把微观领域的量子行为扩展到宏观世界的推演。这里必须要认识影响量子行为的一个现象——观测。这一点非常重要。在经典物理学中，粒子和波是物质（或者能量）的两种不同表现特性。而在量子力学中，微观物质处于一种波粒二象性，即同时具有波和粒子的特性。当我们用双缝实验来观测时，它会表现出波动性；当我们用光电实验来观测时，它又表现出粒子性。

实验是这样的：在一个盒子里有一只猫，以及少量放射性物质（如图 5-10 所示）。放射性物质有 50% 的概率将会衰变并释放毒气，杀死这只猫；同时，放射性物质有 50% 的概率不会衰变，猫将存活下来。

图 5-10　薛定谔的猫

根据经典物理学（宏观低速世界的物理学）的定义，在盒子里必将发生这两个结果之一，而外部观测者只有打开盒子才能知道结果。在量子的世界里，当盒子处于关闭状态，整个系统则一直保持不确定性的波态，即猫生死叠加。猫到底是死是活，必须在盒子打开，外部观测者进行观测时，物质以粒子的形式表现后才能确定。这项实验旨在论证量子力学对微观粒子世界超乎常理的认识和理解，但也使微观不确定原理变成了宏观不确定原理，客观规律不以人的意志为转移，猫既活又死，违背了逻辑思维。

这个实验的过程其实非常烧脑，因为它和我们平常的认知完全不同。猫要么活，要么死，普通人通常无法认知“既是活的，也是死的”这样一种状态。那么，我们该怎么理解这样的试验过程呢？

在这个实验中，“观测”是对实验本身有影响的重要内容。猫在封闭的盒子里，有一定的概率死亡，可是人没有办法获知其状态。也就是说，人类要想描述这个时刻——未打开盒子观察猫的状态，就一定要使用概率。因为根据实验设计的环境，在大量的实验过程中，打开盒子后，有 50%的概率会跳出活猫，还有 50% 的概率会看到猫的尸体。在没有打开盒子的时候，描述猫生死状态的最科学的方法只能是概率，而且这与大量实验的结果一致。

“薛定谔的猫”是诸多量子困惑中具有代表性的一个。刚刚的实验如果换一种方式来实现或许更容易理解：一只猫被封在一个密室里，密室里有食物和毒药，毒药瓶上有一把锤子，锤子由一个电子开关控制，电子开关由放射性原子控制。如果原子核衰变，将放出 α 粒子，触动电子开关，锤子落下，砸碎毒药瓶，释放出里面的剧毒氰化物气体，则猫必死无疑。原子核的衰变是随机事件，物理学家能精确知道的只是半衰期——衰变一半所需要的时间。如果一种放射性元素的半衰期是一天，则经过一天该元素就少了一半，再经过一天就少了剩下的一半，以此类推。然而，没有人能知道原子核会在什么时候衰变——是上午，还是下午？

也就是说，对于一件微观的、不知道具体什么时候发生的事情，我们在宏观上只能观察到用概率解释的事情，在描述中我们使用波函数通常可以体现这种“叠加”状态。甚至这只猫在打开盒子的瞬间以前是“既活又死”的，即在打开盒子的瞬间碰翻了氰化物的瓶子而毒死了猫，由此我们得到了“猫已经死亡”的观测结果，这就是波函数的坍缩。在这个时候，“观测”使实验对象的状态发生了改变。它不仅是实验行为的一部分，而且可以严重影响实验结论的归纳。

但不用担心，如果我们在同一时间对很多只薛定谔猫进行观测，得出的生与死的比例仍旧符合量子力学的概率分布。量子力学中有很多反直觉的怪异现象，但是都符合实验结果。对其原因的解释，学术界存在好几种说法，而最经典的就是哥本哈根学派的解释，我们在前文采用的都是这种解释（这种“一会儿波，一会儿粒子”、“既是波，又是粒子”的说法居然是经典解释）。此外，还有比较小众的“隐变量”解释和比较奇特的“多世界”解释。

看到这种现象，你会担心观测方式将影响归纳的结论吗？我想这种影响是没有办法避免的。

然而，不论影响是什么，我的理解是，只要我们把握一个根本原则——严格记录实验结果，观察到什么就是什么，如果不能精确到个案的预测，那就精确到宏观的概率——这种方式即使不能解开当前的问题，至少不会带来错误的观测结果。

5.5　本章小结

在用数据科学对客观世界进行归纳和认知的过程中，我们无法避免和各种矛盾“短兵相接”，而其中很大一部分是由人类自身认知的局限性带来的矛盾。在“薛定谔的猫”这个实验假说中，我们甚至看到了观测本身左右实验结果的情况，这简直是让人沮丧透顶的结论——只是看了一眼，结果可能就变了。

不过，不知道你有没有这种感觉——认知存在边界本身是有好处的——只要我们的研究与当前的认知水平相匹配，结论与观测结果一致，就可以在现阶段盖棺定论了。至于以后由于认知边界扩展而产生的结论失效，以及对新规则的总结和归纳，可以交给后人去完成。把无限的认知任务化解为有限的认知任务，这同样是解放人脑的重要手段。一好一坏两种矛盾的结果都存在，只要用得好，就不会有无穷无尽的困惑。这难道不是它的好处吗？

第 6 章　实验和哲学

6.1　朴素的认知方法

认知作为人类与生俱来的能力，可以说是上天赐予我们最大的恩惠。我们不仅能感知，还能思考。我们具备令其他生物极为艳羡的智能，即使是世界上最聪明的非人类脊椎动物，以及最先进的人工智能系统，都无法企及。

我们的双眼可以看到周围世界发出或反射的可见光；

我们的双耳可以听到周围世界发出的声波；

我们可以用鼻子的嗅觉闻到花香；

我们可以用舌头的味觉感知酸、甜、苦、咸；

我们可以用双手双脚触及周围世界的一切事物。

应该说，大多数人从来没有怀疑过自己的眼睛看到的一切和耳朵听到的一切，因为它们太“真实”了。

“我看到它，因为它就在那里。”

“我听到它在啼鸣，由远而近。”

我们习以为常地相信，我们感知到的一切就是我们以为的那样。似乎这样的观点也没有什么不妥。换句话说，“感知到”就是存在的标准。

可是，问题马上跟着来了——感知和存在之间能画等号吗？让我们想想——应该不能。

例如，在古代（都不用说远古），哪怕是 500 年前（大约在 1500 年），人们能感知 200 亿光年以外的天体吗？人们能感知到黑洞（如图 6–1 所示）的存在吗？不能，因为没有技术手段。可是，它们存在吗？答案不言而喻。它们的生命周期很多都是以“亿年”为单位的，比人类的历史要长

图 6–1　黑洞

得多——它们当然存在。我们做一个假设，如果有人“穿越”回去，信誓旦旦跟当时的人说世界上存在黑洞这种东西，会有人相信吗？——“你说有，好吧，指给我看看嘛。”面对古人的质疑，现代人要通过什么手段才能让他们知晓呢？恐怕挠破了头也想不出好办法。

6.1.1　眼见为实

中国有句古话：“耳听为虚，眼见为实。”这句话出自汉朝历史学家刘向（约前 77 年 ~ 前 6 年，如图 6–2 所示）的《说苑 · 政理》：“夫耳闻之，不如目见之；目见之，不如足践之。”意思很明白：从他人那里听到的事情，不如亲眼看到的可靠；亲眼看到，又不如亲自去做更贴切。

图 6–2　刘向

刘向的本意是指在治理的过程中，官吏们应该本着实事求是的原则，道听途说不如自己到基层实地查看，仅用眼看不如亲自深入实践。这种精神和自然科学中认知领域“物理学是以实验为本的学科”[44] 的观点是完全一致的。

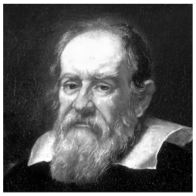


图 6–3　伽利略 · 伽利雷

在这里，“眼见为实”其实已经超过了“眼见”的字面含义。这是一种不要主观妄断，不要通过猜测和推导得出结论，要注重观察与实践的认知观念。在这个方面，世界上最著名的例子可能就是伽利略（Galileo Galilei，伽利略 · 伽利雷，1564 年 2 月 ~ 1642 年 1 月，如图 6–3 所示）和亚里士多德（前 384 年 ~ 前 322 年）关于自由落体速度的“争执”了。那么，两位生活时间相隔 1900 多年的学者是怎么“争执”起来的呢？让我们回到公元前 4 世纪，先看看亚里士多德。

亚里士多德，古希腊人，世界古代史上最伟大的哲学家、科学家和教育家之一，堪称古希腊哲学的集大成者。他一生勤奋治学，从事的学术研究涉及逻辑学、物理学、生物学、教育学、心理学、政治学、经济学等，写下了大量的著作，他的著作是古代的百科全书。亚里士多德的思想对人类产生了深远的影响，在学术届的地位无人能够撼动。

亚里士多德有一个著名的“落体运动法则”理论：物体下落的时间与质量成反比。例如，一物体的质量是另一物体的 2 倍，则在同一下落中只使用另一物体一半的时间。如果空气比水稀 2 倍，则同一运动物体在水中运动时要耗费在空气中运动 2 倍的时间。

这一理论现在看来虽然非常荒谬，但在古代欧洲学术界却被奉若经典。当时这个荒谬的观点仍旧被大多数学者接受并长时间信奉的原因可能有以下 3 个。

第一，测算手段有限。这个理论中提到的“空气比水稀 2 倍”是如何定义的？是密度还是摩擦力？这种说法很难测量并验证。

44　语出杨振宁。

第二，科学观念还不完备。如果科学观念完备，在没有足够观测数据作为证据的时候，就不能认定一个观点或结论成立，更没有道理把它奉为经典——这太荒唐了！

第三，权威压制。在这个理论传承的将近 2000 年里，也许有人曾经对其表示怀疑，至少会质疑其严谨性。然而，对亚里士多德这样有着耀眼光环的学术巨擘，学术界能够容忍质疑的声音吗？这不仅是对亚里士多德的挑战，更是对众多学者信仰的挑战。所以，不论质疑的观点是否科学，在当时的环境中，质疑本身就是一种无形的指责。我不由得想起《理想国》里那个洞中囚徒的例子——明哲保身的学者们或许因此最终保持了缄默。

直到 16 世纪，意大利天文学家、力学家、哲学家伽利略逐步对古代科学典籍进行研究，并对一系列力学实验进行了观测。1589 年 ~ 1591 年，伽利略对落体的运动进行了细致的观察，从实验上和理论上否定了流传千余年的亚里士多德“落体运动法则”，确立了正确的“自由落体定律”——在忽略空气阻力的条件下，质量不同的球在下落时同时落地，下落的速度与质量无关。

根据伽利略晚年的学生温琴佐·维维亚尼[45]的记载，落体实验是在比萨斜塔（如图 6-4 所示）上公开进行的，但在伽利略自己的著作中并未明确说明该实验是在比萨斜塔上进行的，因此，这个著名的自由落体实验故事近年来饱受真实性的争议。但无论如何，伽利略留下的实验结论确实更为科学，也更为准确。这种敢于怀疑经典，并通过科学实验推翻经典的精神，远比否定“落体运动法则”更重要，这也是人类能够不断进步的动力。

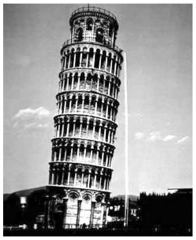


图 6-4　比萨斜塔

6.1.2　归纳与总结

对于生来就处于其中的这个世界，人类从来都不缺乏好奇心。世界上的一切都是我们认知和研究的对象，而且对象永远不会缺少。不论人类是否用量化的方式去衡量这些认知对象，在潜意识中对客观事物进行归纳与总结是千百年来每一代人一直在做的事情。

归纳和总结是相当有意义的过程，通过它我们可以得到平时所说的“经验”。经验是指经历与验证，是一种对事物特性的规律性描述。人们乐于总结经验的原因是什么呢？无非是为了对抗认知中的不确定性。从生产的角度来说，人们相当厌恶具有不确定性的东西，厌恶成本投入中回报这种对象有熵的存在——哪怕这种成本是仅仅是一段时间，而非任何会计师眼中显性的金钱成本。

在农业生产方面就有很多经验的归纳与总结用来指导生产，哪怕是一个没有太高文化水平的农夫，也可以从谚语中获得前人的经验。“天上鲤鱼斑，明日晒谷不用翻（如图 6-5 所示[46]）。”

45　温琴佐·维维亚尼（Vincenzo Viviani，1622 年 4 月 ~ 1703 年 9 月），数学家和物理学家。

46　透光高积云。出现这种情况是因为高气压中心附近的空气向外流散，天气变化少，所以短期内天气不会变化，仍为晴好。

“早霞不出门，晚霞晒死人。”这类人们耳熟能详的谚语能在很大程度上帮助农民进行短期的天气预测，安排生产与抢收工作，为一代代农民增加产出、减少损失提供了最简单也最为基础的经验指导。这些规律，毫无疑问都是通过经年累月对气象变化情况的归纳与总结得出的，也可以说是为了对抗天气变化中的信息熵而产生的。

各个领域中的经验由于体系性不高，往往显得纷繁杂乱。智者们会尝试将这些经验进行再次的总结和归纳。一方面，将那些体系更接近的内容放在一起，进一步酝酿，形成完整的、独立的学科，甚至精确量化这些经验和体系。另一方面，一些人会尝试总结这些看似不在近似体系中的经验和规律背后的规律，这些规律会体现出抽象的、普适的、跨学科的特性，这就是哲学。

图 6–5　天上鲤鱼斑——透光高积云

6.2　哲学靠谱吗

哲学，源于希腊语“*φιλοσοφια*”，从词源上说，有“爱与智慧”的意思。

不论是东方的哲学还是西方的哲学，它们的来源都是庞杂的自然科学，或者是对自然的认知。从大量人与自然互动的过程中，人类逐步进行归纳总结，尽可能得出一些看上去非常宏观、通用（所谓“更有哲理”）的理论和认知观念。

在早期的哲学体系，尤其是西方古代哲学中，宗教是其主要的载体。奴隶社会中后期，随着社会生产力的发展，从事专业研究的科研人员逐渐发生了分化，使数学、天文学、物理学、医学、地理学等学科逐渐从原始的统一哲学中分化出来（如图 6–6 所示）。

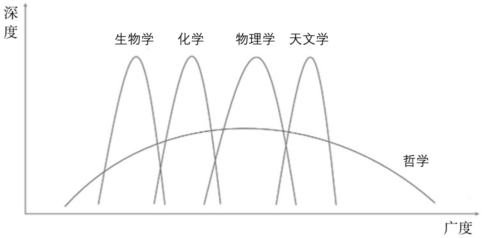


图 6–6　哲学与各学科的深度和广度

有一种说法是“哲学是科学之科学”，这种认识把哲学当成包括一切科学在内的、凌驾于一切科学之上的永恒不变的理论体系。其实，我们稍微想想就知道这种观念可能有问题。因为人类的认知本身就是有限的，人类用有限的认知归纳总结而来的东西就有局限性，也就是说，这种总结出来的东西永远存在例外的情况。无论如何，发展总有例外，那么，我们是不是可以认为例外的东西永远比我们认知到的东西多呢？这可真是个令人无奈到家的结论。

古代哲学家把哲学看成知识的总汇，近代一些哲学家把哲学说成是“科学之科学”。近代以来，随着各门具体科学从哲学中分化出来，哲学和具体科学的关系就成为人们必须正确回答的重要问题。从哲学与具体科学的关系来看，二者是一般与个别、共性与个性的关系。哲学研究的是万事万物的普遍规律，而具体的科学研究的是某一领域、某一方面的规律。

“理论联系实际”是马克思主义哲学中很重要的一条认知原则。这里的“理论”可以理解为数学、物理学等讲述具体实验现象或理论的学科内容。这些内容有一个共性，就是对问题的描述有片面性，是认知维度坍缩到能够清楚描述物质量化关系最小粒度的内容。但是，我们遇到的实际问题通常不是割裂性的，而是很多复杂的因素交织在一起的。我们遇到的实际问题中会夹杂着许多跨学科的因素，如果生搬硬套理论来解决问题，通常会得到和期望的结果相去甚远的不良后果。这是理论研究中维度精简和实际研究中维度爆炸的矛盾给我们带来的困扰。

6.3 数学的尽头是哲学

前面我们提到过这样一句话：“物理学的尽头是数学，数学的尽头是哲学，哲学的尽头是神学。”要想更好地理解这句话，不妨站在数学这个点向“两边”看。

先看左边。所谓“物理学的尽头是数学”，是指通过物理实验总结了大量定量的体系结构和模型，建立了测量值之间的逻辑关系。要想描述这种关系，就需要使用数学这种简洁的、忽略具体场景的、只研究共性模型的描述语言。

再看右边。所谓“数学的尽头是哲学”，是指数学是哲学的一种表现形式。数学和哲学都是较为抽象的学科，而且都是研究学科共性及万事万物普遍规律的。数学还有一个优势就是定量化描述，这个优势哲学可没有。

在《自然哲学的数学原理》[47]（如图 6–7 所示）中，牛顿通过“物体的运动”、“物体在阻滞介质中的运动”和“宇宙体系”（使用数学的论述）3 编对自然物体和天体运行（运动）的模型进行了证明和建模。在自然物体和天体运行中有大量与直线、圆、椭圆、抛物线、双曲线方程描述一致的运动模型，这是实实在在地用数学语言来解释自然运动的规律。如果你不认为其中有哲学存在，那我们换几个数学表达式来看看。

$$\lim_{n\to\infty}\sum_{i=1}^{n}9\cdot(0.1)^{i}=1$$

这个表达式其实等价于 $0.\dot{9}$——一个无限循环小数。它等于多少？等于 1。要注意，这里可不是“约等于”，而是“等于”。这样一个表达式就是在表现一种量变到质变的思想。可以说，这是用数学表达式表达哲学观点的一种形式。

47 *Mathematical Principles of Natural Philosophy*，成书于 1687 年。

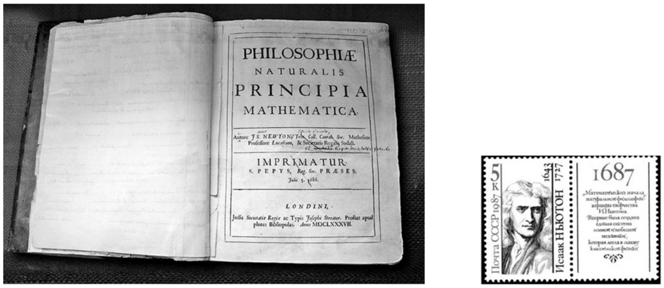


图 6–7　英文版《自然哲学的数学原理》和其出版 300 周年的纪念邮票

《汉书 · 艺文志》中对诸子百家有这样的评述：“其言虽殊，譬犹水火，相灭亦相生也。仁之与义，敬之与和，相反而皆相成也。”这里的“相反而皆相成也”就是一个哲学观点，是指两个对立的事物既互相排斥，又互相促成，即相反的东西也相互依赖，具有同一性。

正数和负数就是一对可以相互依赖和转化的矛盾。

$$(-1)\times(-1) = +1$$

2 个 –1 相乘就转化成了 +1。

$$(+1)\times(-1) = -1$$

+1 作为正数，在乘以 –1 的情况下就转化成了 –1。这种表达式表现了对立事物的转化。

当点 (x, y) 在方程

$$x^2+y^2 = 1$$

所表示的曲线上匀速运动时，x 和 y 都在 –1 到 +1 之间进行往复运动——周而复始，物极必反。这种表达美轮美奂，而且在任何国度之间无须翻译即可沟通，真是妙哉。

6.4　本章小结

关于实验和哲学的纷争其实已有定论。

哲学是跨学科的通用性规律的归纳和总结，但是针对性差，而且由于局限性问题，实际上需要不断自新才能保持健壮性，这一点与其他自然科学并无二致。

而实验则不同，与单纯的数学或哲学相比，它在特定的环境中更容易受具体维度的影响。实验作为科学的、实事求是的验证载体，更是成为归纳和总结科学理论及哲学的证据性行为，是最应该受到重视的科学实践过程。所以，哲学应该做好对自然科学内容的沉淀；而实验，尤其是精确测量的实验，才是认识世界的根本手段。

第 7 章　辩证思维

约公元前 5 世纪，古希腊哲学各派论争之风盛行，哲学家们都比较注重争论的技巧和方法。这时，人们把论证或分析命题中的矛盾，以及在谈话中揭露对方论断中的矛盾并克服这些矛盾以求得真理的方法，叫作辩证法。

人们通过概念、判断、推理等思维形式对客观事物辩证发展过程的正确反映，即对客观辩证法的反映。辩证思维最基本的特点是将对象作为一个整体，从其内在矛盾的运动、变化及各个方面的联系中进行考察，以便从本质上系统地、完整地认识对象。

辩证思维注重的是从多个角度看待事物，推敲事物认知中的矛盾，进而得到一个统一的概念或解释。如果不进行辩证思维，而是片面地认知事物，通常会发现认知中有很多矛盾，也就是不能自圆其说的地方。

7.1　要不要辩证有多大区别

《两小儿辩日》（如图 7-1 所示）是一则脍炙人口的文言文寓言，选自列御寇[48]所编的《列子·汤问》，原文是这样的。

孔子东游，见两小儿辩斗，问其故。

一儿曰："我以日始出时去人近，而日中时远也。"

一儿以日初出远，而日中时近也。

一儿曰："日初出大如车盖。及日中则如盘盂，此不为远者小而近者大乎？"

一儿曰："日初出沧沧凉凉，及其日中如探汤，此不为近者热而远者凉乎？"

孔子不能决也。

两小儿笑曰："孰为汝多知乎？"

文章大意如下。

图 7-1　两小儿辩日

48　列御寇，战国时期郑国圃田（今河南省郑州市）人，道家学派的代表人物，著名的思想家、寓言家和文学家。

孔子向东游历，见到两个小孩儿在争辩，就问他们争辩的原因。

一个小孩儿说："我认为太阳刚升起的时候距离人近，而到正午的时候距离人远。"

另一个小孩儿认为，太阳刚升起的时候距离人远，而到正午的时候距离人近。

一个小孩儿说："太阳刚出来升起的时候大得像车盖，到正午就像圆盘一样大，这不是远的小而近的大吗？"

另一个小孩儿说："太阳刚出来的时候很清凉，到中午的时候就像把手放进热水里一样烫，这不是近的热而远的凉吗？"

孔子不能决断。

两个小孩儿笑着说："谁说你见多识广啊？"

简简单单的一则寓言故事，给我们讲述了一个非常有趣的辩论过程。可以说，在古代认知水平有限的情况下，人们同样拥有辩证的智慧，即使是三尺顽童也不例外。在这段对话中可以看出，虽然两个小孩儿的结论是完全矛盾的，但他们都引用了非常有力的证据对自己的观点进行论证。这种现象在日常生活中屡见不鲜，而且有趣的是，两个人对"中午热、早晚凉"这一客观现象的认知一致，却有着矛盾的解释。

在观测手段发达的今天，《两小儿辩日》中的这种温度问题已经不再是问题了。有一些地理或天文常识的人都知道，《两小儿辩日》中争论的太阳和地球的距离是没有太大变化的，引起温度变化的主要原因是太阳光里短波辐射的变化。太阳在正午的时候直射地球，地球吸收的太阳光辐射能量最大，所以温度高；太阳在早晚的时候斜射地球，地球表面吸收的太阳光辐射能量较小，所以温度低（如图 7–2 所示）。这种差别和地球上的寒带、温带、热带的气温特性原理成因是非常相似的，都是因为阳光照射角度影响了地面对太阳辐射能量的吸收程度。

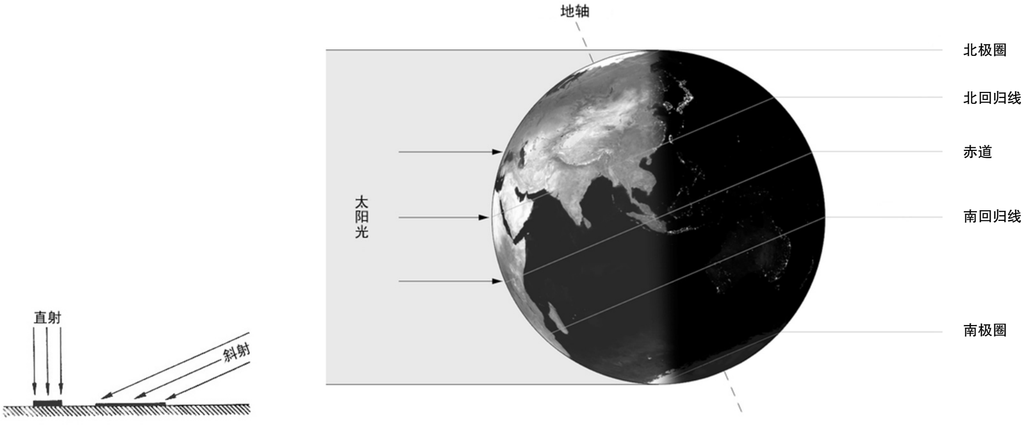


图 7–2　太阳光对地面的直射和斜射

孔子作为著名的思想家、教育家，如果说他没有能力决断两个孩子的辩论是没有智慧，这恐怕有失公允。孔子不能决断的原因是由他的认知局限性导致的，因为在当时，这些对光和地球自转与公转的认识，以及对辐射与内能的认识等，都尚未建立。这种情况下，真是让圣人很为难。

辩证思维在这里看上去好像有点“巧妇难为无米之炊”的意思，可这并不是辩证思维的错，因为无论如何，哪怕是像故事中讲的这样——三个人都疑惑不已，也比直接得到像“太阳早晚近，中午远”或者“太阳早晚远，中午近”这样的错误结论要好得多。

7.2 谁对谁错

在平时的工作和生活中，我们免不了要对一个问题进行反复讨论并做出决策，这个过程看起来像是在判断一件事情或者一个观点的对错。在不可避免的争论中，双方各执一词的情况居多。这并不奇怪。事物虽然是客观存在的，但是每个人从自己的角度看到的内容却很可能是不同的——即使看到的内容相同，在脑海里投射的印象也可能不同；即使在脑海里投射的印象相同，基于不同目的做出的反应也仍旧可能不同（如图 7–3 和图 7–4 所示）。

图 7–3　是人脸，还是杯子

图 7–4　是美女的脸，还是吹萨克斯的男人

这丝毫不奇怪，趋利避害本身就是人和其他生物所具备的一种本能。而我们通常看到的恰恰是反应最为明显和激烈的最后一个环节，那就是“基于不同目的做出的反应”。这样就不难理解为什么对于同一件事情或者同一个观点，会有这么多迥异的态度，而且争执起来喋喋不休。

即使是我们看上去已经很科学的数学也无法幸免——并不是在讨论课题时运用了数学原理或科学测量的手段就能避免争执。据我所知，在很多大的公司或团体中，一个课题通常是由多个专家或者专业分析团队来合作解决的。每个专家、每个团队都有相当强的分析能力，他们解决问题时的出发点和评价体系都不完全相同，这种不同直接导致思路的不同，也会影响课题的解决成本和解决效果。通常这些团队之间的讨论也不会有什么结果，最多就是让他们彼此知道别的团队是怎么想的、怎么做的，以及以什么评判条件作为目标。但是，只要众多团队所确定的目标不明确，一般无法得到一种公认的合理解决方案。

谁对，谁错，这重要吗？或以为是，或以为非。

如果要在工作或学习中，或者任何其他环境里，在处理问题的时候一定要得到一个极为确切的评价结果，只有一条路——严格地定义一个没有二义性的数字化标准，以及一系列对这个

标准负责任的没有二义性的数字化评价体系。如果不建立这个标准，无论讨论多久都不会得到能够服众的结果。

就拿国人议论了 30 多年的高考来说，“应试教育”、“八股考试”都是扣在高考头上的“帽子”。虽然社会上绝大部分人都赞成“素质教育”这种说法，最后走向工作岗位看素质，高考就学也应该看素质，可“素质”是什么？如何比较不同学生的“素质”？你会弹钢琴，我会 110 米栏，哪个更高明？还有那些拥有其他特长，以及在某一专业有造诣的学生，谁应该获得进学资格？这种尝试不限制评选维度的想法只要一冒头，其提出者本人都会立刻发现，没有标准的评选结果将直接导致相互说服的时间成本无穷大——没有标准，不能在统一的维度上做量化，结果可想而知。

想来想去，结果是把高考的竞争方案简化为比较考试的总分，这就是一个把所有因素统一到一个量化维度上进行比较的方案。即使再做修改，也只是改变组成或换算“总分”这个量化维度的方式，无非就是调整参与评分的科目及各科目的分数权重等。

这个方案最公平吗？未必公平。因为要讨论是否公平，要想服众，同样要有一个没有二义性的对“公平”的定义，并且可以没有二义性地量化它。然而，一万个人心中有一万种“公平”的定义，所以，从这个角度来说，“公平”没有标准。不过，只要有一个普遍基本认同的标准，在改变它之前直接去比对这个标准下的量化值就可以了。这是一种决策过程中性价比非常高的方式，具有很高的实践价值。

7.3　做到客观不容易

在很多情况下，我们都会强调：“要客观！要客观！”客观认识事物，客观评价事物，但是客观真的太不容易实现了。“客观”是一个抽象名词，意思是在意识之外，不依赖精神而存在的，不以人的意志为转移的。“客观”是事实存在的，和“主观”对立。

可是我们同样知道，人类对世界的认识都是通过人类的主观感知获得的——通过视觉、听觉、嗅觉、触觉、味觉等方式来感知，然后由人的大脑记忆。以上这些环节有一个算一个，本质就不“客观”。

人们想出的补救办法就是用数字化的精确的仪器来感知，然后以数字形式来记录。一般情况下，用仪器进行测量确实会精确很多，但是仪器在解决客观认知问题的过程中能够提供的帮助有限。为什么？因为最后一级的认知和判断工作还是要由人来完成，只要是人参与的事情，就很难做到百分之百“客观”。一起来看下面的例子。

眼睛是人体非常宝贵的器官，它不仅能感知光线的强弱，还能感知光线的不同波长。光线从人的瞳孔射入，最后成像于视网膜（如图 7–5 所示）。只要人的眼睛基本健康，就能够清晰成像并感知光线的强弱和波长。对波长的感知表现为对颜色的感知（如表 7–1 和图 7–6 所示）。

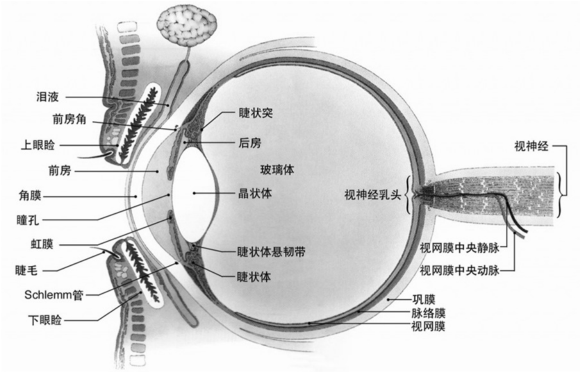


图 7–5　人眼的结构

表 7–1　不同波长光线的颜色

颜　色	波长（纳米）
紫色	380~450
蓝色	450~475
青色	475~495
绿色	495~570
黄色	570~590
橙色	590~620
红色	620~750

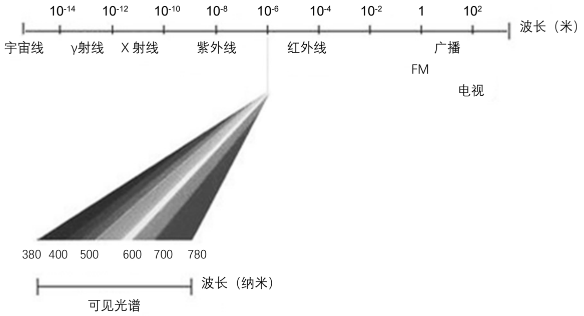


图 7–6　不同波长的电磁波

正常的人眼可以辨识 1000 多万种颜色。我们说的“辨识”，不是指能够说出“深红”、“浅蓝”这种名称。如果把两种不同的颜色放在一起，人类能够感知其差异，就叫作能辨识这两种不同的颜色。

在数字图像处理领域，通常使用 RGB（也就是红、绿、蓝）3 种颜色通道对颜色进行描述和组合，每个通道有 256 种取值，这样 3 个通道最多可以表示 16777216 种颜色，即我们平时说的“24 位真彩色”。对颜色高度敏感的人甚至能识别比这个数量更多的颜色，也就是对光波长的细微变化敏感。但是，也有一些人对颜色感知的能力要弱一些，他们可能只具有对不到 1000 万种颜色的辨识能力。

由于基因方面的问题，一些人在出生的时候就会罹患一种叫作“色盲症”的疾病。现在通常说的“色盲”大多是指红绿色盲。在人的视网膜上有一种感光细胞叫作“锥细胞”，它有红、绿、蓝 3 种感光色素，每种感光色素主要对 1 种原色光产生兴奋，并对其他 2 种原色光产生程度不等的反应。如果缺乏某种感光色素，就会产生对此种颜色的感觉障碍，表现为色盲或色弱（辨色力弱）。

色盲可以分成许多类型，仅对 1 种原色缺乏辨别力者称为单色盲。红色盲，称为“第一色盲”，比较多见；绿色盲，称为“第二色盲”，人数比“第一色盲”少一些；蓝色盲，称为“第三色盲”，在 3 种之中人数最少。如果对 2 种或 2 种以上的颜色缺乏辨别力，称为“全色盲”，这种较为罕见。色盲多为遗传所致，少数为视路传导系统障碍所致。在基因层面，一般是女性传递，男性表现。根据统计，男性色盲发病率为 5%，女性则为 1%。有先天性色觉障碍者，往往不知道自己的辨色力异常，多为他人觉察或体检时发现。凡从事交通运输、美术、化学、医药等工作的人，必须有正常的色觉，因此，色觉检查就成为服兵役、就业、入学前体检时的常规项目（如图 7–7 所示）。

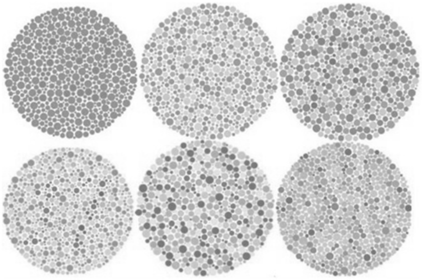


图 7–7　色盲检测卡

我们来做一个极端的假设，如果不是像我们刚刚讲的那样——男性色盲发病率为 5%，女性为 1%，而是男女色盲比例都有 99.99%，例如认不出绿色，会发生什么事情？或许会变成这样。

所有的数字电视都只有红和蓝 2 个通道，一共有 65536 种颜色。彩虹的颜色也不是我们说的七彩了，肯定缺少绿色，甚至对其他颜色的辨识也会受到影响。交通信号灯会设计成“红灯停，蓝灯行”。而剩下的 0.01%，即那些能够完整辨识 16777216 种颜色的人，会被称为“拥有特异功能的人”。这些“特异功能”人士之间谈论的各种有关绿色的话题“常人”就完全听不懂了。“常人”只能通过精密的仪器才能感知这些只有“特异功能”人士才能感知的颜色所表现出来的波长示数与他们眼中的可见光有所不同。而在比较系统的论述色彩与波长之间关系的理论出现之前，是很难有“常人”得出这种结论的——“常人”根本感知不到，何来研究？

事物本身是客观的，但感知是主观的。将主观的感知用于客观的事物，这种叠加还能保证客观吗？这真的很难服众。如果一个人与周围大部分人甚至是世界上大部分人的认知结果一致，它就是客观或者真实的吗？还真不一定。

7.4　观念的存弭

万事万物都有一个“从生到死”的过程，这里说的“生”和“死”不是指生物的“生”和“死”，而是一种“存”与“灭”的过程。从无到有，从有到强，从强到衰，从衰到灭，万事万物都逃脱不了这样一个规律。

历史上曾经出现过无数的科学理论，这些科学理论虽然在当时也被认为是科学的，但也没能逃脱这种淘汰规律。究竟在什么情况下这些科学理论会被淘汰，最终消弭在历史的漫漫长河中呢？答案也很明确：只要拥有辩证思维的智慧，科学理论的代谢就只取决于感知的边界扩张了。

具体一些，能对一种观念的“生”和“死”起重要作用的就是新认知维度的引入，而这才是引起观念存弭的最重要的因素。

人类出现在地球上之后，能够凭借眼睛、耳朵和手感知到的内容是非常有限的，能够使用双脚走过的地方也是非常有限的。人类看到的天穹是圆的，大地是一个平面。这可不是一个人的认知，而是所有的人都看到了，这时说“天圆地方”是没什么问题的。所以，这时如果有人说“大地是个球体”，那他就会被认为是疯子。

公元前 6 世纪，毕达哥拉斯通过观测月球表面的地球的投影猜测地球应该是球状的，当时很少有人认同这一观点。即使到了 1522 年，葡萄牙人麦哲伦（Fernando de Magallanes，费迪南德·麦哲伦，1480 年～1521 年 4 月）的船队完成了人类历史上第一次环球航行，并向全人类证明了“大地是个球体”这个说法时，仍然有很多人感觉非常意外（如图 7–8 所示）。大地是个球体？那下面的人怎么办？会不会掉到太空里去？这是很多对“大地是个球体”表示诧异的人最为“忧虑”的事情。与其说是忧虑，不如干脆说是不解其奥妙。直到牛顿发现了著名的“万有引力定律”，这个问题才算有了完整和确切的科学解释。根本不存在什么“球体的上和下”，所有的人和物体都是被“吸附”在地球表面的。牛顿同时提出了“牛顿三大定律”来量化解释宏观低速物体的力、质量与速度的关系——真是太精妙了，所有人都觉得这几乎精确解释了宇宙中所有的运动规律，直到出现了爱因斯坦这个“疯子”。

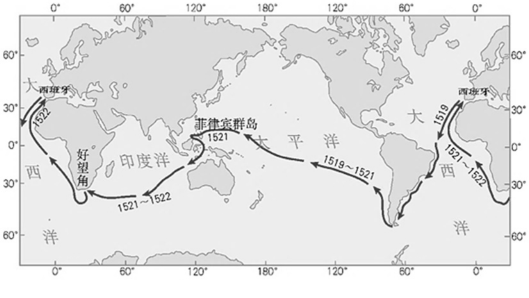


图 7–8　麦哲伦航海路线

爱因斯坦 1905 年 9 月发表在德国《物理学年鉴》（*Annalen der Physik*，如图 7–9 所示）上的那篇著名的相对论论文《论动体的电动力学》中，提到了有关光速的问题。

光在空虚空间里总是以一确定的速度 V 传播着，这速度同发射体的运动状态无关。

下面的考虑是以相对性原理和光速不变原理为依据的，这两条原理我们定义如下：

物理体系的状态据以变化的定律，同描述这些状态变化时所参照的坐标系究竟是两个在互相匀速平行移动着的坐标系中的哪一个并无关系；

任何光线在“静止的”坐标系中都是以确定的速度 V 运动着，不管这道光线是由静止的还是运动的物体发射出来的。

对于大于光速的速度，我们的讨论就变得毫无疑义了；在以后的讨论中，我们会发现，光速在我们的物理理论中扮演着无限大速度的角色。

由此，当 $v = V$ 时，W 就变成无限大。正像我们以前的结果一样，超光速的速度没有存在的可能。

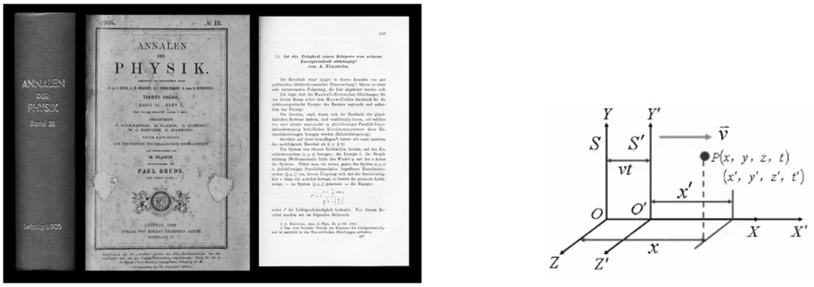


图 7–9　1905 年的德国《物理学年鉴》和四维空间的洛伦兹变换示意图

这种论调在当时的物理学家看来简直就是彻头彻尾的疯狂，尤其是经典力学的信徒兼牛顿的“铁粉”们，简直愤怒到无以复加的地步。可是光愤怒不解决问题，一切都要以观测到的内容为准。

根据光速不变原理，无论在何种惯性系（惯性参照系）中观察，光在真空中的传播速度都是一个常数，为 299792.458 千米/秒。

证明光速不变的 4 项事实如下。

（1）恒星光行差。

（2）恒星都是一个一个的小圆点。

（3）恒星都静止。

（4）太阳光迈克尔逊–莫雷实验（如图 7–10 所示）。

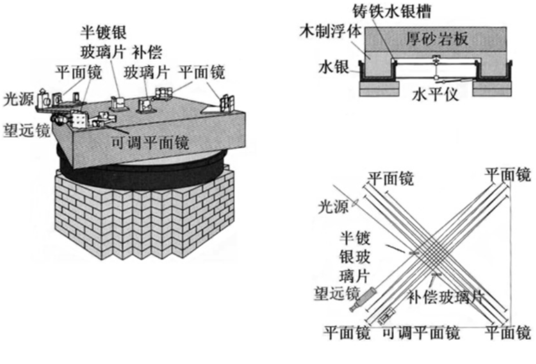


图 7–10　迈克尔逊–莫雷实验示意图

其中，迈克尔逊–莫雷实验不仅证明了光速不变，而且证明了许多物理学家一直相信的使得光波进行传播的绝对惯性系“以太”实际上并不存在。实验原理非常简单：光波长和频率都是由光干涉条纹确定的。根据“杨氏双缝干涉实验”干涉条纹之间的距离，能够独立推算出“光波长”，自然可确定“光频率”。

这样推算确定的光波长和频率的乘积为常数，即不同颜色光的波长和频率的乘积相等，而且乘积等于检测的“光速值”，从而充分证明

$$\text{光速} = \text{波长} \times \text{频率}$$

是成立的。

迈克尔逊和莫雷通过长期实验，分别多次检测来自不同方向的阳光的光速，充分证明：阳光的光速不变。

既然光速为 299792.458 千米/秒，如果真的有一个人以 299792.457 千米/秒的速度运动，并用手中的手电筒向他运动方向的前方发出一束光，那么光对他来就是以 1 米/秒的速度运动。这时要想让整个系统继续有完整的认知体系作为支撑，只能让时间变得极慢，几乎静止——时间被压缩为只有原来的 $\frac{1}{299792458}$。这些概念，别说是当时的人，就算是对近现代习惯认知宏观低速物理环境的人来说，都是不可思议的事情——真是一代一代的“疯子”把人类的认知抬高了一层又一层。

这些认知结论会不会在不久的将来又一次被其他“疯子”推翻呢？“将来”多久会来不好说，只从认知理论的科学观念来看，可能性很大。而当有另一个“疯子”提出新的发现和理论解释时，如果我们是科学认知的虔诚信徒，就应该用科学的态度对待这样的事情，用科学的评价标准来评价这些新的认知成果。

7.5 本章小结

我们在“认知篇”里举了很多的例子来说明认知的重要性，以及认知中的迷茫。困惑层出不穷，从人类有文明记载开始到现在，几千年来一直在持续。它让我们感到痛苦吗？不。恰恰相反，正是有了这些迷茫的存在，才让人类拥有了能够不断去挑战的认知边界。正是在这种不断对认知边界的挑战中，人类发展出了物理学、化学、生物学等近现代学科，还发展出了更多的诸如核物理、量子物理、基因学、量子化学等尖端学科。人类文明就是在这种不断的挑战和进步中获得了一次又一次质的提升。

在整个“认知篇”的最后，我想用一个小故事来回答以前屡屡被同行朋友问起的问题：一个数据科学家有没有必要多读一些数据科学之外的读物？我想答案是肯定的，不仅有必要，而且很有必要。

这个故事收录在袁枚[49]（如图 7–11 所示）的《随园诗话》卷四。清乾隆七年（1742 年），袁枚在应天府[50]上元县[51]当县令，五月初十那天，上元县突然发生了可怕的龙卷风，家住县城的韩宗梁家里 19 岁的女儿韩慧芬在外面收衣服时被龙卷风卷走，不见踪影。后来发现，她竟被吹到了 80 里外的江宁县铜井村[52]。被发现时她坠落在地，不省人事，铜井村人只好用姜汤将她灌醒。韩慧芬睁开眼时，周围尽是陌生人，互相询问才得知此地为铜井村。因天色已晚，身体有伤，她便在铜井村一户人家住了一晚，第二天才被送回城里。

图 7–11　袁枚像

韩慧芬回来后，对众人诉说这件奇事。大家一听，觉得很蹊跷。韩慧芬的未婚夫李调甘听说此事，也觉得不可思议，认为其中一定有隐情——在那时，发生如此暧昧的事情，这门亲事结不得了。于是，李调甘来到上元县衙击鼓告状，要求退婚。

袁枚接案后，先是听了李调甘的诉词，接着传来韩家人询问事情的经过。为了深入调查事实，袁枚又到江宁县，把铜井村的地保传到县衙作证。地保说，当日大风起，确有一铜井村村民，姓华，名中元的，在家门外突然看到一阵黑气由北飞来，到他家门口时落下。开始以为是龙取水，仔细看才发现一女子躺在自己家门前且已经昏倒。该女子经过大家的救治活了过来，询问后得知她是城里韩家的女儿，因天色已晚留宿一宵，第二天送其回家，余情不知。

经过多方讯问，大家供述无异，案情清楚：韩慧芬到铜井村，确系风吹所致，也确实由铜井村人救活，因为天晚路远，当天不便送回，才留宿一宵，均为常事，没有异常。而城里那天确实发生了龙卷风，也毁坏了不少房屋，邻居市民可以为证。再看韩慧芬，端庄本分，不像水性杨花之人。可是无论袁枚怎么说，迂腐的李调甘就是不相信，总是怀疑另有隐情。

这时，袁枚只有回过头来劝解李调甘："你也是读书人，应该见多识广，江南发生龙卷风是春末夏初常有的事，古书上就有女子被风吹走 60 里的事。"说着，袁枚取来元朝名臣郝经[53]的著作《陵川集》[54]，找到了一首记载此事的诗，念给李调甘听：

八月十五双星会，花月遥光照金翠。
黑风当筵灭红烛，一朵仙桃落天外。
梁家有子是新郎，芋氏负从钟季背。

49　袁枚（1716 年 ~ 1797 年），汉族，钱塘（今浙江杭州）人，清朝乾嘉时期代表诗人、散文家、文学评论家。
50　现江苏省南京市一带。
51　现江苏省南京市江宁区一带。
52　现江苏省南京市江宁区铜井一带。
53　郝经（1223 年 ~ 1275 年），元初名儒，政治家，学者。
54　元代文人郝经作品集，今存三十九卷，是流传较广的元人文集之一。

争看足下来鬼物，云鬓欹斜倒冠佩[55]。
须臾举目视旁人，衣服不同言语异。
自说吴门六十里，恍惚不知来此地。
甘心肯作梁家妇，诏起高楼榜天赐。
几年夫婿作相公，满眼儿孙尽朝贵。
须知伉俪有姻缘，富者莫求贫莫弃。

袁枚念完诗，继续说道：“郝经是元朝的一代名臣，他记载了‘风吹人’事件，难道他会说假话吗？你看，那次被风吹走女子嫁的人后来做了宰相，我只怕你还没有那么大的福分呢。不要再提退婚了，回去完婚吧！”李秀才这才转怒为喜，表示愿娶韩慧芬为妻。

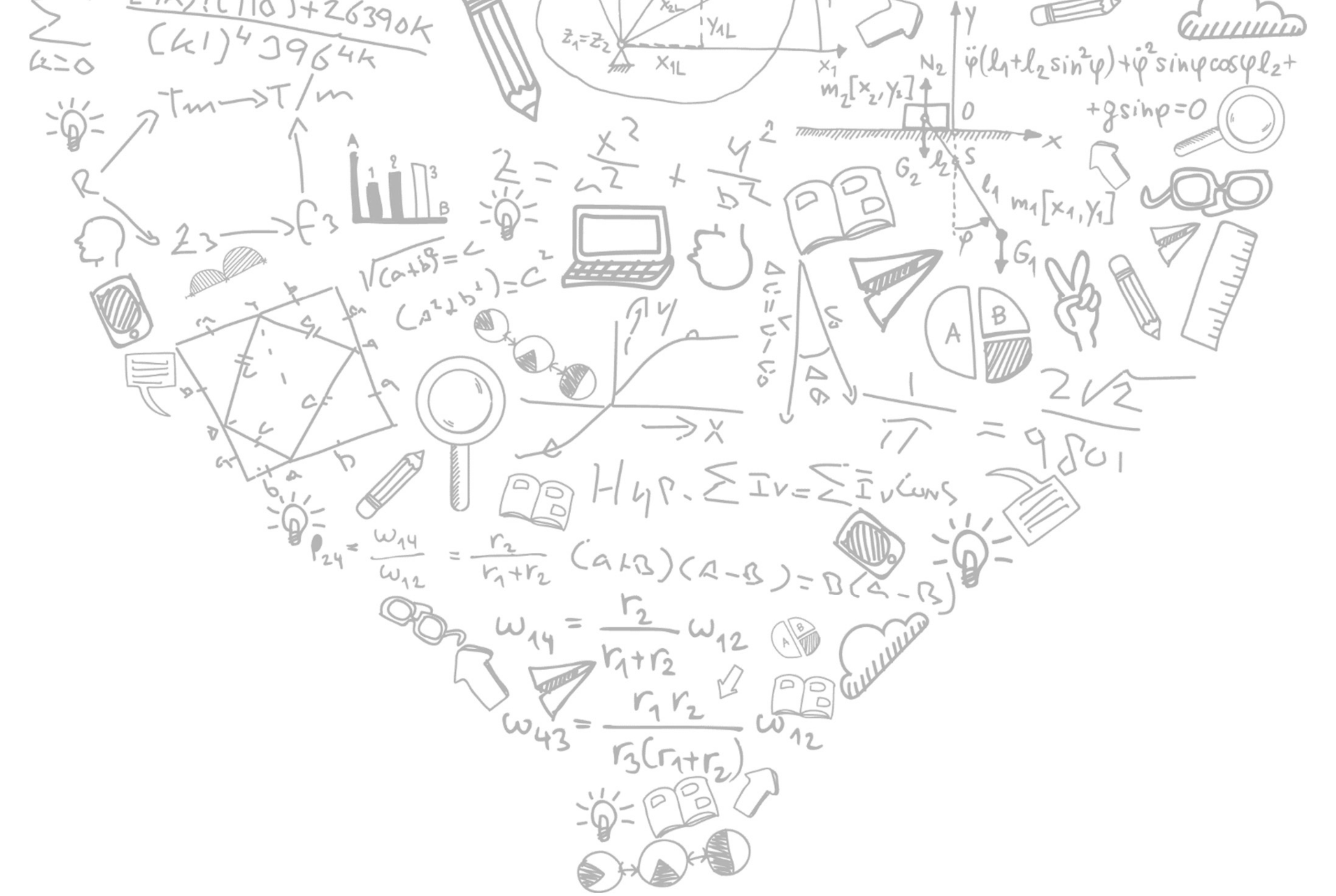
于是，袁枚提笔判道：“本月十日，却有狂风。走石飞沙，毁屋伤人。韩慧芬起而收衣，被风卷走；铜井村凭空落下，惊为天人。众人救助，护送回城。个中细节，俱已详审。合情合理，没有疑问。无端猜忌，冤屈好人。天下之大，何奇不有。风能吹人，史有明文。读书五车，身入黉门[56]。当局者迷，旁观者清。本官再三开导，秀才撤诉息争。疑团一一冰释，两家重续前盟。姻缘本是前定，无妨委屈；好女不是杨花，岂怕风吹；今日就备花烛，本官赠你鼓吹。全两美之情，平双方之恨。即日成礼，夫妇齐眉。朝朝暮暮，两心如一；恩恩爱爱，永结百年。此判。”

这个故事到这里就结束了。是不是离奇且有趣呢？

县令袁枚在断案的过程中，通过对自然现象的认识和对古籍的引用，成功地帮助未婚小夫妻断了一桩蹊跷的案子。案子的过程听起来确实让人匪夷所思，用现代科学一点的说法，应该叫“小概率事件”。被龙卷风袭击的人，生还概率通常极低，但是在浩瀚的古籍读本中，袁枚还是找到了有人生还的证据。也就是因为这样一个飞来的证据，让一个没头没尾的糊涂案有了破解的方法。这与清朝律法没有丝毫关系的只言片语却能够帮助县令断案，是不是就相当于我们在做数据分析的过程中酌情使用行业知识、业务特性甚至生活经验和人生经历呢？扩展知识面本身就非常有助于在数据分析过程中拓展思路，也便于找到真正的矛盾，进而聚焦问题，破解问题。

亲爱的读者朋友，你还在犹豫什么呢？

55 欹，读音同“漆”，倾斜，歪向一边。
56 黉，读音同“红”，古代指学校。



分化篇

在人类认知逐渐发展的过程中，由于感知和观测手段的进步，使很多逐步可以聚合成一个垂直学科的分类，从混沌的哲学思维中分离出来，成为一门独立的学科。这种分化过程是认知进步的表现。

第 8 章　统计学

统计学是一门非常重要的学科，也是现在大数据应用中最为基础的学科，是数据分析的重中之重。在合理应用统计学理论的基础上，通常我们已经能够解决日常生产中绝大多数的数据释义问题。

统计学之所以称为科学，是因为它是一门体系完备的学科，数据计算模型完整，数值计算准确。通过现在已经普及的数据库技术，对数字进行计数、求和、求平均等运算的成本越来越低。每个企业、每个单位都希望对手里掌握的原始数据进行梳理、总结、提炼，不论是为了指导运营，还是为了向上报告，都需要有一个数字化的总结。那么问题来了：统计工作过程怎样才算没有问题呢？有问题的话，问题有多大呢？怎样进行统计更为科学呢？

背诵经典书籍的定义是没有意义的。要想做好统计这类工作，还是要理解工作的实质内容，而且上升一个维度和层次进行理解，效果会更好。下面，我们就把统计学中最为基础且关键的问题说个清楚。

8.1　数理统计鼻祖——阿道夫·凯特勒

阿道夫·凯特勒（Adolphe Quetelet，1796 年 ~ 1874 年，如图 8-1 所示），比利时统计学家、数学家、天文学家、物理学家，被誉为“国际统计会议之父”、“近代统计学之父”、“数理统计学派创始人”，主要著作有《论人类》、《概率论书简》、《社会制度》和《社会物理学》等。

图 8-1　阿道夫·凯特勒

凯特勒出身于比利时甘特市的一个小商人家庭。1819 年，23 岁的凯特勒在甘得大学获得博士学位。1823 年，凯特勒建议政府建立天文台。为了天文台的筹建工作，他被派往法国学习。由此，他与拉普拉斯、普阿松、傅里叶等人相识，并从师拉普拉斯学习概率论。1827 年，凯特勒游学英国伦敦。1829 年 ~ 1830 年，他先后到德国、法国、瑞士、意大利等国考察。据说，他因为偶然接触了人寿保险公司的实际业务问题，开始对统计学进行研究。1823 年，天文台建成后，凯特勒被任命为台长，并开始发表人口及犯罪方面的统计研究。1841

年，比利时中央统计委员会成立，凯特勒任终身主席。

凯特勒发现，那些表面上似乎杂乱无章的、偶然性占统治地位的社会现象，如同自然现象一样，也具有一定的规律性。他认为，统计学不仅要记述各国的国情，研究社会现象的静态，而且要研究社会生活的动态，以及社会现象背后的规律性。凯特勒的这一思想为近代统计学的科学化奠定了基础。他还认为，社会现象背后的这种规律性是社会内在固有的，而不是“神定秩序”，人们可以通过计算统计指标来揭示这些规律。凯特勒的这些思想给后世统计学家带来了深刻的影响。

犯罪统计中所呈现出来的规律性，竟使凯特勒联想到司法机构的经费预算问题。1829 年，凯特勒作出了这样的论述：“可预想每年有同一犯罪以同一序列重复出现。监狱和法院的预算，与国家每年的收入几乎同样确定。”1835 年，他在《论人类》一书中又说：“世界上，人们每年按某一惊人的常例来确定用于监狱、徒刑场和断头台等开支的预算。虽然人们想尽力节约这笔开支，但只要仔细考察这些开支数目，却发现每年都不幸中了我的预言。”对于凯特勒的上述成就，马克思曾给予充分的肯定：“凯特勒先生在 1829 年发表的对可能出现的罪行的估计，不仅以惊人的准确性预算出了后来 1830 年法国发生的犯罪行为的总数，而且预算出了罪行的种类。”凯特勒根据统计资料得出结论：鉴于最贫穷地区的犯罪数目不及经济发达地区的犯罪数目大，因此，犯罪反而与经济（走向）富裕有关。

凯特勒的上述工作，处处闪烁着他社会统计规律性思想的光辉，给后人以极大的启迪。凯特靳还将法国的古典概率引入统计学，用纯数学的方法对社会现象进行研究。

凯特勒运用概率论的方法进一步研究了社会道德领域的大量统计资料，发现了以下基本原则：“在我们对多数人进行观察的时候，人的意志就平均化起来，并且不留任何显著的痕迹。所有部分意志的作用，和纯粹受偶然原因所制约的各种现象一样，它们即被中和或抵消了。”这就是凯特勒著名的“平均人”思想。他认为，“不应当注意个别的人，而应当把个别的人当作种族的一部分来考察。只有把人的个性去掉之后，我们才能把存在于人们中间的所有偶然的东西摒弃殆尽。这样，那种对于大量现象仅起极小作用的、或完全不起作用的个别特殊性，就自然会平均化起来，从而我们就能把握住综合的结果”。同时他还认为，对社会上偏离“平均人”的差异性，也要研究其发生的原因。据他研究，社会上所有的人与“平均人”的偏差愈小，社会上的矛盾也就愈缓和。而文化上的正面引导，则可以减少每个人与“平均人”的偏差，从而减少犯罪的发生。凯特勒的“平均人”思想在历史上影响很大。马克思在《资本论》中也曾运用这种思想。孔子的“不患寡而患不均，不患贫而患不安”[57]也这是这种思想的体现，只不过孔老夫子没能把这个概念量化而已。

凯特勒就是这样在自己的研究工作中把统计学与概率论结合起来。他首次在社会科学的范

57　出自《论语・季氏》第十六篇。

畴内提出了大数律思想，并把统计学的理论建立在大数律的基础上，认为一切社会现象也受到大数律的支配。他的这种统计思想曾盛行一时，至今还有影响。

1857 年，凯特勒在第三次国际统计会议上论证了概率论方法对于统计价值测定的必要性。1867 年，他在第六次国际统计会议上提出，希望建立一个特别小组委员会来处理直接与概率论有关的统计问题。凯特勒不仅把概率统计的方法引入人口、领土、政治、农业、工业、商业、道德等社会领域，还把概率统计的方法引入天文、气象、地理、动物、植物等自然领域。他的这种关于概率统计的方法，是应用于任何事物数量研究的最普通的思想方法，对以后统计学的发展具有重大意义。[58]

8.2 统计就是统共合计

统计学究竟为我们解决了什么问题？为什么要设置这样一个分支领域呢？

最早的统计学可以追溯到 17 世纪英国的政治算术，代表人物是威廉·配第（William Patty，1623 年 ~ 1687 年，如图 8–2 所示）。他在《政治算术》[59]一书中写道，本书“不用比较级、最高级进行思辨或议论，而是用数字来表达自己想说的问题，借以考察在自然中有可见的根据的原因”。政治算术学派主张用大量观察和数量分析等方法对社会经济现象进行研究，为统计学的发展开辟了广阔的前景。

图 8–2 威廉·配第

除此之外，还有国势学派、社会统计学派和数理统计学派。

在学科性质上，凯特勒认为，统计学是一门既研究社会现象，又研究自然现象的方法论科学。从 19 世纪中叶到 20 世纪中叶，数理统计学得到了迅速发展。

英国生物学家高尔顿[60]提出并阐述了相关的概念。皮尔逊[61]提出了计算复相关和偏相关的方法。戈塞特[62]建立了“小样本理论”，即所谓的“t 分布”。费歇尔[63]在样本相关系数的分布、方差分析、实验设计等方面的研究中作出了重要贡献。到 20 世纪中期，数理统计学的基本框架已经形成，数理统计学派成为英美等国统计学界的主流。经过 300 多年的发展，目前统计学已经成为横跨社会科学和自然科学领域的多科性的科学。

统计学是有关如何测定、收集和分析反映客观现象总体数量的数据，以便给出正确认识的

58 摘自互动百科，有删节。
59 *Political Arithmetick*，1676 年出版。
60 弗朗西斯·高尔顿（Francis Galton），1822 年 2 月 ~ 1911 年 1 月。
61 卡尔·皮尔逊（Karl Pearson），1857 年 3 月 ~ 1936 年 4 月，英国数学家、生物统计学家，数理统计学的创立者，自由思想者。
62 William Sealey Gosset，1876 年 6 月 ~ 1937 年 10 月，英国统计学家。
63 罗纳德·费歇尔（Ronald Aylmer Fisher，1890 年 ~ 1962 年），英国统计学家。

方法论科学。从横向看，各种统计学都具有上述共同点，因而能够形成一个学科“家族”。从纵向看，统计学方法应用于各种实质性科学，与它们相结合，产生了一系列专门领域的统计学。由此可见，现代统计学可以分为两大类：一类是以抽象的数量为研究对象，研究一般的收集数据、整理数据和分析数据方法的理论统计学；另一类是以各个不同领域的具体数量为研究对象的应用统计学。

关于统计，对学过这门课但没有深入进行系统性研究的人，以及只是听过这个名字但对此一知半解的人来说，似乎就是算算总数，算算平均值，然后做出报表，用表格和图表对数字进行展示。从日常生活中的应用形式上看，是这个样子不假，然而，很少有人会在这个过程中用科学的方法去实践。统计学究其本质，还是希望通过科学且简洁的方法对数量巨大的群体进行特征归纳与描述，并可以在小范围的试验中以很高的精确性反推全样本的性状。

最为简单的统计方式就是数量的计算。例如，“今天到场一共 45 人”，“本次央行量化宽松的额度为 4 万亿元人民币”，“2014 年中国人口数量为 13.68 亿人”，这样的陈述就是统计数字。但这不是统计学，这些单纯的孤立数字还没有上升到学科的高度。统计学有清晰的边界及对研究对象的定义。

总体（Population）是包括所研究的全部个体（数据）的集合，它通常由所研究的一些个体组成，例如多个企业构成的集合、多个居民户构成的集合、多个人构成的集合等。每一个组成总体的元素称为个体。在由多个企业构成的总体中，每一个企业就是一个个体；在由多个居民户构成的总体中，每一个居民户就是一个个体；在由多个人构成的总体中，每一个人就是一个个体。总体的范围在一些场合下是容易确定的，在另一些场合下是不容易确定的。在有些场合，总体的数量比较小，在一次实验中可以全部覆盖；而在有些场合，总体的数量极为庞大，无法通过一次实验全部覆盖。

样本（Sample）是从总体中抽取的一部分元素的集合。构成样本的元素的数目称为样本量（Sample Size）。抽样的目的是根据样本提供的信息推断总体的特征。例如，从一批灯泡中随机抽取 100 个，这 100 个灯泡就构成了一个样本，可以根据这 100 个灯泡的平均使用寿命去推断这批灯泡的平均使用寿命。这种方式在日常生产中使用非常广泛，极大地降低了质量监控的成本，而且对不同批次产品的质量有严格且精确的描述能力。这种“一叶落知天下秋”[64] 的哲学思想是统计学在抽样统计一环具体体现出来的优秀特质。

参数（Parameter）是用来描述总体特征的概括性的数字度量，是研究者想要了解的总体的某种特征值。研究者所关心的参数通常有总体平均数、总体标准差、总体比例等。在统计中，总体参数通常用希腊字母表示。例如，总体平均数用 μ 表示，总体标准差用 σ 表示，总体比例用 π 表示。

64　出自《淮南子・说山训》。

应该说，“统计”二字就是“统共合计”的意思，是一个期望用简洁的表达方式对大量信息进行归纳和抽象的分支科学。而在统计过程中，对总体、样本、参数 3 个核心特性进行把握，往往能得到很好的实验效果。

8.3 数据来源

要想进行统计，首先要确定统计对象，也就是前面我们提到的总体。在这之后，就是从总体中把数据收集上来。

常用的数据来源有两种。第一种是间接来源，也就是由他人通过调查或实验的方式收集的数据，使用者只是找到它们并加以使用。第二种是直接来源，即通过自己的调查或实验，直接获得第一手数据。

间接来源的数据一般来说会有很多第三方渠道提供，例如政府统计部门、专业调查机构、专业期刊文献、学术会议资料库提供的数据等，也包括一些单位自己收集并整理的营业记录、消费记录等。只要不是统计分析的实施者亲自获取的数据，都应该算作间接来源的数据。

间接来源的数据一般来说获取成本比较低，不过局限性也比较大，主要存在的问题有统计口径不一致、时效性不强等，所以在使用前需要对这些数据进行评估，例如数据提供方的信誉度、数据的收集手段、数据的收集时间等，从而最终判定这样一份数据是否可以作为统计分析的对象。

直接来源的数据，其获取成本通常比较高。在很多情况下，当间接来源的数据无法解决我们要研究的问题时，也只能考虑使用直接来源的数据，就是通过调查方法获得数据，或者通过实验方法获得数据。前者获得的数据叫作调查数据，后者获得的数据叫作实验数据。不论是调查数据还是实验数据，要想获得它们，都需要设计一套严谨的实施过程，从调查、数据产生、数据收集到数据分析，都需要有严格且科学的解释作为保障。

调查通常以一定范围内的社会人群为总体，实验通常针对具有自然性的现象。基本可以这样认为：需要人对类似答卷作出主观性回答的形式（即使是选择题）都可以称为调查；实验则是对观测现象进行相对客观的记录（例如，读取温度计的读数，以及对实验中符合某种状态的对象进行数量记录等）。在互联网和物联网产品中，通过终端收集的数据通常可以算作实验数据。

数据不论是间接来源还是直接来源，一定要建立在足够可信的基础上。如果对数据的信任程度不够，不建议盲目地开展分析，因为分析结果通常会变得不可靠。这里指的不是误差，误差是客观存在的，也无法避免。这里说的是数据来源、统计口径、记录方式等的可信度问题。数据来源就是我们所说的“认知”的第一步，是作为分析者观测的开始一环。不知道你是否能意识到，这一环中的数据解释说明和数据可靠性判断是统计能够正常进行的基本保障。在解读数据之前，要确保其可靠性能够满足我们对解读精度的需求。

8.4　抽样

通常情况下，把总体全部作为样本进行调查研究的成本是极为高昂的。在对大量的人口对象进行统计的时候就有这个问题。对几十亿条数据进行多维度的统计分析，这样的计算，即使是用计算机来进行，也有无法忽略的成本。此外，检测本身可能存在破坏性。例如，在工业品出厂检测中，很多情况下，被测对象在被拆解后无法再流向市场，这就形成了一种生产过程中的损失。所以，有相当多的统计调查都是通过抽样来进行的，因为这种方式更为经济，而且可行性更高。

为了让抽取的样本对总体有足够好的代表性，在生产中，最常使用的方法就是概率抽样（Probability Sampling），也叫作随机抽样。例如，使用正 N 面体的匀质骰子进行抽样，使产生每个标号数字的概率相同（如图 8-3 所示），这种方式广泛地应用于生产检验环节、博彩业及其他领域。除此之外，还有分层抽样（Stratified Sampling）和非概率抽样（Non-Probability Sampling）等多种抽样方法。

图 8-3　20 面骰子和 6 面骰子

抽样的目的是用少量的数据样本来代表总体，也就是说，希望通过比例性的扩大反推抽样集合中样本的特性在总体中的量与分布。所以，抽样通常被认为是随机性越高就越理想，最好能够避免对某种特征样本的集中抽取——除非实验设计的研究对象就是这些特征样本。

在日常生活中，如果要实现这种尽可能随机的抽样，可以使用匀质骰子或者计算机语言中的随机函数。抽取的对象通常是类似员工 ID 的尾号、手机号码的尾号，以及其他一些在生成过程中没有经过过多人为干预的散列特性很好的序列。

8.5　对照实验

有了数据来源，有了抽样方法，还需要一些方式来帮助我们识别认知对象的差异和差异来源，这就用到了对照实验。对照实验是一种统计研究的方法，在互联网领域和其他涉及大量服务对象的行业中，会倾向于使用一种叫作“AB 测试”的对比方式，其实就是统计学中的对照实验。

对照实验的思想方法很简单。将观测对象分为 A 和 B 两个组，A 组称为“实验组”，B 组称为“对照组”。在实验过程中有意识地改变某个变量（例如变量 x）的情况，然后观察另一个变量（例如变量 y）变化的情况。对实验组实施一个动作，而对对照组不实施这个动作。一段时间后，对比实验组和对照组中的个体对象是否体现出差异，就是刚刚说到的变量 y 的差异是否存在。一般来说，两个组在整个实验过程中的环境变化要保持一致，以尽可能消除其他因素带来的影响。

例如，在药品的临床测试中，要测试一种新药是否真的有效，或者其效果是否比其他药品

的效果好，通常会采用“随机对照试验”（Randomized Controlled Trial，RCT）。RCT 将病患分为两组：如果要验证药品是否有效，就一组给药，另一组不给药；如果要对比两种药物的效果，那就一组给待测药，另一组给一种同类对比药，在疗程结束后对比治愈率。为了避免人为的情绪化因素及个别样例的特殊反应对测试结果的影响，又进化出一种叫作“大样本随机双盲试验”的方法，算是对 RCT 进一步科学化的诠释。

大样本随机双盲试验的特点如下。

第一，大样本。样本量加大会稀释个别样例的特殊反应对统计结果的影响，这种做法就是统计学中所说的“避免过拟合”。

第二，双盲。病人和医生对药品和分发对象事先不知情，也就是说，病人和医生都不知道某位病人服下的是待测药还是对比药，所有的人都在这个被他人安排好的测试中一步一步进行试验，直到最后才去对比测试的结果。这就避免了在人与人接触的过程中由于主观情绪造成的一些难以把握的因素。例如，医生如果主观上对某种药的效果不看好，或者认为某种药的效果不错，在交流过程中就会或多或少地对病人进行情绪上的暗示，这种暗示对病人配合治疗程度的影响会干扰测试结果——说不清究竟是药品有效，还是其中有更多“安慰剂”[65]的成分。双盲测试同样是为了保证刚刚所说的“实验过程中的环境变化要保持一致，以尽可能消除其他因素带来的影响”。

早在 17 世纪初，英国海军就试图运用实验的方法找到坏血病的成因。当时，长期在海上航行的水手们面临着坏血病的威胁，如果得了这种病，皮肤上会出现青灰色的斑点，牙龈会大量出血。1747 年，詹姆斯・林德[66]（如图 8-4 所示）做了一很著名的实验，让 12 位患有严重坏血病的海员在航行中食用完全相同的食物，唯一不同的是，他们会分别服用当时传说可以治疗坏血病的“药物”。其中，2 位海员每天吃 2 个橘子和 1 个柠檬，2 位海员每天喝苹果汁，其他海员分别每天服用稀硫酸、酸醋、海水或其他当时认为可以治疗坏血病的药物。6 天之后，只有吃橘子和柠檬的 2 位海员病情好转，其他人病情如前。1748 年，詹姆斯・林德以优异的成绩获得爱丁堡大学医学博士学位后，回到海军任军医。

图 8-4　印有詹姆斯・林德头像的纪念邮票

在这项实验中，吃橘子和柠檬的 2 位海员构成了实验组，喝苹果汁和服用其他“偏方”的海员构成了对照组。需要对照组的原因是，如果没有对照组，就

65　安慰剂（Placebo）具有一定的作用，对有心理因素参与控制的自主神经系统功能（例如如血压、心率、胃分泌、呕吐、性功能等）的影响较大。它所产生的心理效应对病症的缓解，在临床上已经得到了相当程度的认可。

66　詹姆斯・林德（James Lind，1716 年 10 月 ~ 1794 年 7 月），英格兰卫生学的创始人，皇家海军外科医生（1739 年 ~ 1748 年），皇家海军哈斯拉医院医师（1758 年 ~ 1783 年）。

无法判定刚才我们所说的变量 x 是否对变量 y 产生影响。一个好的实验设计都由 1 个实验组及 1 个或多个对照组组成。这种方法广谱且有效，是一种很好的认知方式。

如果我们要回到 1747 年去帮助英国海军设计这个实验，同样也要注意，3 个对照组的对象应该尽量随机，例如在实验过程中应该考虑他们对饮食的偏好。在实验组和对照组中，都应该放入喜欢与不喜欢吃橘子和柠檬的两类人，而且数量应该相同。在这种环境下，起码可以避免饮食喜好问题对疗效的影响。

一个好的实验，对照组和实验组的产生不仅应该是随机的，而且应该是匹配的。所谓匹配是指对实验单位的背景材料进行分析比较，也就是像我们刚刚假设的那样，将情况类似的每对对象分别随机地分配到实验组和对照组。例如，在前面提到的大样本随机双盲试验中，会尽量让接受实验的病人按照年龄、性别、病情等变量进行匹配，然后将他们分别分配到实验组和对照组中。这样，不同组中的病人就有大致相同的背景。

但是，我们必须要注意一个问题，那就是：实验的理想程度越高，实施的成本也就越高。这是一个实验可信程度与成本的平衡性问题。在一个实验中，对照组样本的“背景”究竟是一个什么样的范畴，这个问题需要我们认真考虑。在刚刚说的詹姆斯・林德所做的橘子和柠檬对坏血病影响的实验中我们已经发现了端倪，这个所谓的“背景”就是关于对实验结果有影响的参数的选择问题。例如，是否应该让 2 组水手尽量保持体重、身高、家族病史等的分布相近。因为，不选择这个参数作为背景进行随机分布的前提就是：我们认定这个参数对实验结果的影响可以忽略不计，否则实验结果其实是不可信的。

我们来说一个极端甚至有些荒诞的例子。在一个实验中，设计有 A 组和 B 组分别作为实验组和对照组，A 组 10 人全部为 40 岁以上男性，B 组 10 人全部为 40 岁以下女性，然后观察 2 组对象是否生长胡须。在实验记录中记下：“A 组 10 人，40 岁以上；B 组 10 人 40 岁以下。经过观察，A 组全部长胡须，B 组全部没有长胡须。”这会不会得出“40 岁是人长胡须与否的分水岭”这样的结论？

无论出于什么原因，实验结果的记录者漏记了 2 个样本组的性别。虽然记录的内容是“准确无误”的，可是最终结果却荒谬到家了。要注意：漏记参数和在无意中没有把参数选入“背景”导致的结果很可能一样可怕。问题是：应该把什么样的因素考虑进去呢？

这个边界的划定在我看来是一个“性价比”的问题。学术上虽然没有确切的定义，不过在实际操作中，我们显然会优先考虑放入那些可能对统计结果产生影响的因素及确定成本较为廉价的因素，这个原则应该是没有问题的。

当然，这种事情同样可以通过其他方式避免。在以统计学为理论基础的机器学习的知识体系中，有一个环节叫作“Validation”（验证），也就是把实验结论中的限制条件附加在另外一些观察对象上，看看是否能得到与实验相同的结论，从而检查其应验性。如果能够得到同样的结

论，就说明这种实验结论的可靠性是值得信任的。这个环节非常重要，也是确保这种归纳认知过程科学性的一种关键手段。对照实验是一种非常广谱的实验方式，对那些维度过于复杂的认知对象来说，对照实验是通过压缩差异维度的数量来提升认知能力的有效手段。

8.6 误差

误差作为一个客观存在的现象，从我们意识到这个世界的存在时就已经开始了。在我们用某种量具进行数值测量的时候，误差已然存在，哪怕是再精确的量具都无法消灭误差，只能尽可能让误差接近零。这就是理想和现实的差异。

误差有两种，分别是抽样误差和非抽样误差。

8.6.1 抽样误差

抽样误差（Sampling Error）是指由抽样的随机性引起的样本结果与总体真值之间的误差。由于抽样本身是一种用少量样本“代表”总体的行为，所以无法避免“以偏概全”的情况发生。也就是说，基于抽样的分析对基于总体的分析来说是一种样本缩减的次优解，牺牲的是抽样误差，得到的是统计计算效率。我们只要从这种角度来看待这两种方案的取舍与代偿性就好。

不仅是在同一次的抽样与总体之间会产生统计差异，不同的抽样批次之间也会有差异。

假设在 10000 个零件中有 1000 个残次品，我们随机抽取 100 个进行合格率检验。第 1 轮抽取的 100 个零件中有 10 个残次品，残次率为 10%；第 2 轮抽取的 100 个零件中有 11 个残次品，残次率为 11%。然后，进行第 3 轮、第 4 轮……大部分的样本结果会落在一个区间内。

在这里，我用 Python 编写了一个脚本，对实验过程进行模拟。

```
import random
population = [0 for i in range(10000)]
for i in range(1000):
  population[i] = 1
for iterator in range(10000):
  sample = [0 for i in range(100)]
  for i in range(100):
    sample[i] = population[random.randrange(10000)]
  count = 0
  for i in range(100):
    count+=sample[i]
  print '%.2f' % (float(count)/100)
```

以上代码模拟了 10000 次从有 1000 个残次品的 10000 个零件中每轮抽取 100 个零件的过程，最后输出的值可以拿来做分析。下面示意性地给出输出内容的前 10 行。

```
0.10
0.11
0.17
0.08
0.09
0.15
0.07
0.10
0.08
0.09
```

将每次抽样的结果进行归纳（如表 8-1 所示）。其中，“0.01”是指抽取到残次品的比率，代表“100 个产品中有 1 个残次品”，这样的抽取结果出现了 5 次；有 17 次抽取到 2 个残次品；有 49 次抽取到 3 个残次品……以此类推。

表 8-1 数量与分布

比 率	轮 数	比 率	轮 数	比 率	轮 数
0.01	5	0.09	1204	0.17	110
0.02	17	0.10	1347	0.18	67
0.03	49	0.11	1205	0.19	23
0.04	163	0.12	995	0.20	14
0.05	358	0.13	723	0.21	6
0.06	579	0.14	553	0.22	1
0.07	891	0.15	312	—	—
0.08	1181	0.16	197	—	—

我们把这个结果画成一幅图（如图 8-5 所示），横坐标是比率（0～0.22），纵坐标是轮数，很明显能看出这是一个正态分布的情况。有 95% 的轮数都取在中心 0.1、左右各 0.06 的范围内。在 0.1－0.06 到 0.1＋0.06 的范围内，和全部残次品真实值 0.1（10000 个零件中有 1000 个残次品）来比，0.06 就是抽样误差。

这个实验很简单，但仍然说明了一个问题：这种随机误差在一轮抽样中其实是很难发现的。例如，在某轮抽样中抽到了 100 个产品，发现其中有 22 个残次品，那么这 22% 的残次率究竟和真值相差多少？要想找到真值的位置，需要进行多次抽样。

在抽样会产生抽样误差的情况下，通过多次抽样才能找到真值，这确实让人很头疼，因为多次抽样增加了计算的成本。所以，在现在计算机硬件价格越来越低的情况下，很多公司喜欢用全部样本直接进行统计也是情有可原的。如果仍旧希望通过多次小样本评估来推测全体的分布情况，可以使用 8.8.3 节介绍的卡方分布方法。

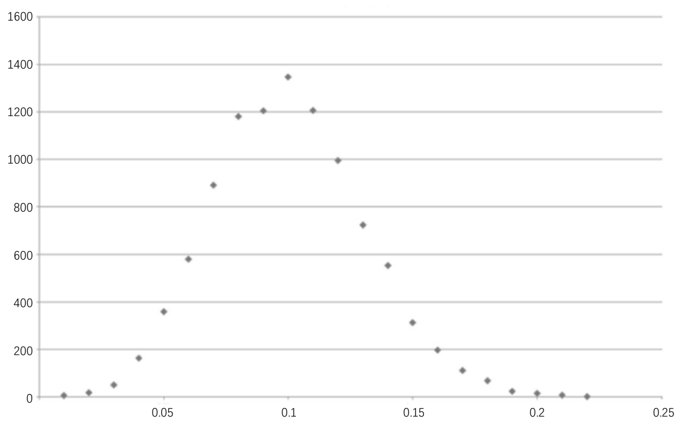


图 8-5 数量分布

8.6.2 非抽样误差

造成非抽样误差的原因就比较复杂了，有很多种情况。

1. 抽样框误差

在对动态事物进行观察研究的情况下都会出现抽样框误差。例如，在规定时间内进行全国人口普查，最理想的情况是在某一时刻对全国人口进行“快照”，将这一瞬时值的状态“冷冻”起来进行普查。但是，这种“冷冻”其实根本不可能实现，也就不可能在规定的时间和地点范围内实现真正的“快照”，而在动态中产生重记和漏记的可能性就会存在。这就是抽样框误差。

这种“快照”方式只能对那些相对容易做快照的系统数据进行统计。很多用批处理操作生成报表的工具都是这样，在第二天的凌晨处理前一天记录的数据信息，并形成关于前一天的快照统计信息。这种方法是有效的，不过其代价是对时效性的牺牲。

2. 回答误差

回答误差在对人的访谈中最容易出现。

回答者对问题的理解是否与提问的初衷相符？回答者是不是理解了问题？这都不容易确定。在回答问题的时候，回答者需要进行回忆，而这种回忆是否准确也不容易确定。还有一些问题，例如回答者由于羞愧、虚荣、道德压力等原因给出了违背其自身真实想法的答案，这也属于误差。这种误差在一些统计手段中有相应的修正技巧，在 16.6.2 节会进行介绍。

3. 无回答误差

无回答误差可以直接理解为“访谈对象交白卷”。在机器收集数据的过程中，会由于各类故障而生成空白的数据记录，这同样属于无回答误差。此外，还有由于调查人员操作不当而产生

的漏记、错记，由于误导回答者而产生的误差，以及由量具和测量方式产生的测量误差等。

对无回答误差，在学术层面有很多方法可以进行代偿。例如，对人来说，可以要求回答者重新作答；对机器报送中发生的失误，可以在机器修复之后用时序上前后连续的部分进行数据平滑等。不过，目前还没有能够完美地避免无回答误差的方法。

一般来说，我们认为在一次调查中，持续时间越久，因素越复杂，人为干扰成分越多，非抽样误差存在的可能性就越大。如果希望减小误差，还是要采用多次测量、去噪声点、取平均值等手段。

8.7　概括性度量

费歇尔[67]在 1952 年的一篇文章中举了一个例子，说明如何由基本的描述统计量知识引出一个重要的发现。

20 世纪早期，哥本哈根卡尔堡实验室的施密特（J. Schmidt）发现，在不同地区捕获的同种鱼类的脊椎骨和腮腺的数量有很大的不同，甚至在同一海湾的不同地点捕获的同种鱼类也发现了同样的倾向，但是鳗鱼的脊椎骨数量变化不大。在从欧洲各地、冰岛、亚速尔群岛和尼罗河等几乎分离的地点捕获的鳗鱼样本中，施密特通过计算发现了几乎一样的均值和标准差。

在派出一艘名为“Dana”的科考船进行多轮捕捞和数据比对以后，施密特判断：各个海域的鳗鱼是在某个公共场所繁殖的，这个地方基本可以圈定在西大西洋东北部和西印度群岛的北部（如图 8–6 所示）。

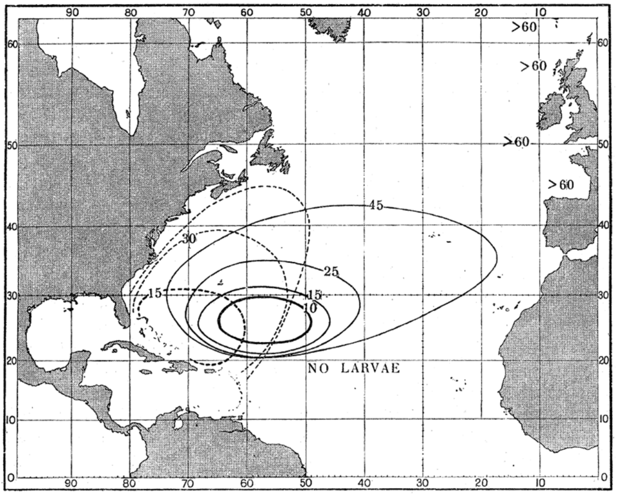


图 8–6　鳗鱼繁殖场的位置

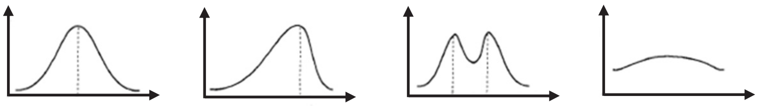
67　费歇尔（Ronald Fisher，1890 年 ~ 1962 年），英国统计与遗传学家。

这种从多批次大量数据中找出共性的方式，并不是人类所擅长的，因为人类的认知能力对少量的对象比较敏感，也比较容易集中精力，而对大量的对象则通常难以进行有效的认知。所以，归纳与概括大量数据的指标就出现了。

8.7.1 集中趋势度量

描述数据集中程度的数据指标有很多，例如众数、中位数和四分位数。

众数（Mode）是一组数据中出现次数最多的变量值，我们习惯用 M_0 来表示。众数主要用于测度分类数据的集中趋势，也适合作为顺序数据及数值型数据集中趋势的测度值（如图 8–7 所示）。一般来说，数据量较大的时候众数才有意义。



(a) M_0 (b) M_0 (c) 双众数 (d) 无众数

图 8–7 众数分布示意图

看这样一个例子。一个小区的理发师，在对当天所有前来理发的 8 位顾客做了年龄登记后，得到一个年龄列表“15、20、22、22、23、35、50、72”，其中 22 就是众数。众数反映的是一个多数的概念，即一个数字比其他数字出现的次数更多或者更普遍。在“15、20、22、22、23、35、50、72、72”中有 2 个众数——22 和 72。

但是，众数的问题也显而易见：如果在众多不同的数字中出现 2 个相同的，这个相同的数字作为众数来表示“普遍”的含义，就与表达的初衷有很大的不同。因此，生产统计中，众数的应用并不普遍，或者需要搭配其他统计指标才能让受众获得比较完备的数据概括描述。

中位数表示一组数据排序后处于中间位置上的变量值。

$$\text{中位数位置}=\frac{n+1}{2}\ (n\ \text{为数据的个数})$$

也就是说，如果有一组数据 $x_1,x_2,\cdots,x_n$ 按照从小到大的顺序排列，新序列是 $x_{(1)},x_{(2)},\cdots,x_{(n)}$，则有

$$\text{中位数}=\begin{cases} x_{(\frac{n+1}{2})} & n\ \text{为奇数} \\ \frac{1}{2}\{x_{(\frac{n}{2})}+x_{(\frac{n}{2}+1)}\} & n\ \text{为偶数} \end{cases}$$

四分位数的定义和中位数差不多，相当于用中位数对排序后的数列进行一次“分割”，再在中位数两侧的 2 个子数列里各进行一次中位数的“分割”。从理论上讲，有中位数和四分位数的定义，就可以有八分位数、十六分位数，以及所有自然数的 N 分位数。然而，对于描述一组数字的集中样态来说，这些“分割点”有些多余，虽然描述得更为细致，但是显然不够直接。所

以，在大多数的日常生产场景中，极少使用这些分位数。

此外，描述数据集中程度的数据指标还有平均数和加权平均数，它们也是统计学中通常用来描述集中趋势的度量值。

平均数也叫均值（Mean），就是一组数据相加后除以个数得到的结果，是目前日常统计中应用最广泛的一种统计指标值。

加权平均数（Weighted Mean）是根据分组数据计算的平均数。例如，原始数据被分成 k 组，各组的平均值分别是 $M_1, M_2, \cdots, M_k$，各组变量值的数量分别是 $f_1, f_2, \cdots, f_k$，则样本加权平均数的计算公式为

$$\bar{x} = \frac{M_1 f_1 + M_2 f_2 + \cdots\cdots + M_k f_n}{f_1 + f_2 + \cdots\cdots + f_k}$$

或者写作

$$\bar{x} = \frac{\sum_{i=1}^{k} M_i f_i}{n}$$

从形式上来看，后者更简洁一些。

掌握众数、中位数和平均数的含义和特点，有助于在实际应用中选择合理的测度值来描述数据的集中趋势（如图 8-8 所示）。

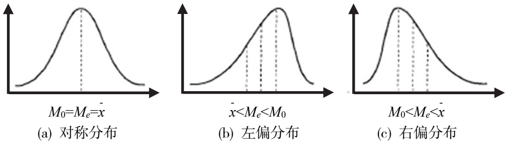


图 8-8　对称分布、左偏分布和右偏分布

众数是一组数据分布的峰值，不受极端值的影响，但是可能不唯一，而且数据量较少的情况下不适合使用。

中位数是一组数据中间位置上的代表值，也不受数据极端值的影响，适合作为顺序数据集中趋势的测度值。

平均数针对数值型数据进行计算，而且利用了全部数据信息。当数据呈对称分布或接近对称分布时，3 个代表值相等或接近相等。平均数的主要缺点是容易受到极端值的影响，对偏态分布平均数的代表性较差。所以，在偏斜程度较大的情况下，使用中位数和众数比使用平均值的描述效果要好。

如果不确定分布状况，或者希望将数据集中趋势描述得完备一些，可以在统计中同时使用这 3 个值，这样就可以通过这 3 个值的大小很快对分布是否对称及如果不对称将呈现哪种偏态

有完整的印象。简单地说，众数、中位数、平均数三者结合使用，可以简洁而形象地描述一组数据的集中趋势，建议在日常生产中组合使用。

8.7.2 离散程度度量

对于一组数据离散程度的度量也有很多常用的指标，例如方差、标准差、极差。

方差公式为

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

方差的平方根 s 称为标准差。

从方差的定义就可以看出，样本值离平均数 $\bar{x}$ 越远，s^2 就越大，样本就越分散。相反，最极端的情况就是所有样本的值都是平均值 $\bar{x}$，那么方差为零，这种情况下样本集中分布。

标准差是平方差的平方根，同样具有描述离散程度的性质。

在实际应用中，平方差和标准差的效果差不多。只是要注意，在一次实验中最好只使用平方差或标准差中的一种来描述离散程度，以免由于量纲混乱而无法比较，进而给人造成困惑。

极差（Range）是指一组数据中最大值与最小值之间的差，用 R 来表示。

$$R = \max(x_i) - \min(x_i)$$

极差只能描述一组数据中 2 个极值的距离，对于中间值的分布状况没有办法描述。所以，这个指标所包含的信息量不大，在日常统计中的使用机会也不多。

8.7.3 小结

概括性度量是统计中最为基本的度量方式，包括集中趋势度量指标和离散程度度量指标两大类。在日常生产生活中描述大量数据样态的时候，尤其是在数据量极大的情况下，进行全数的罗列毫无效率可言，而且让人抓不住重点。应该使用众数、中位数、平均数这种集中趋势度量指标与方差这样的离散程度度量指标相结合来进行描述，这样做通常可以达到简洁和使数据描述信息丰满的双重目的。比较多组不同数据之间的差别时也可以使用这些指标，不仅结果准确，而且能提高数据的处理效率。这就是使用概括度量的意义所在。

8.8 概率与分布

随着数据科学的普及，“概率”这个词从原来仅出现在教科书和科技文献中，慢慢成为人们挂在嘴边的一个常用词汇。例如，在天气预报中就经常提到“降水概率”这个词。

概率的定义其实是一个比例问题，我们用最简单的例子来说明。有一枚匀质的硬币，硬币有两面，一面为字，一面为图案。将硬币掷出去落地以后，可能是字的一面朝上，也可能是图

案的一面朝上。字的一面朝上，就称为一个事件，我们称其为事件“正”；图案的一面朝上，也称为一个事件，我们称其为事件“反”。

事件“正”和事件“反”属于随机事件，即在同一条件下，每次试验中可能出现，也可能不出现的事件。

一种事件出现的可能性的大小称为概率。例如，事件“正”的概率记作 P(正)。

概率有几种不同的定义，分别是古典定义、统计定义和主观概率定义。

古典定义的概率是指在随机试验结果有限的情况下，某种事件与所有事件的个数比。需要注意的是，古典定义所说的概率有两个共同点：第一，结果有限，即基本空间中只含有限个元素，例如刚刚说的事件“正”和事件“反”（只有 2 个）；第二，各个结果出现的可能性被认为是相同的，也就是在掷硬币的情况下事件“正”和事件“反”出现的机会被认为是相等的。

在古典定义下，可以使用排列组合进行概率估算。但我们知道，这种情况的局限性是非常大的。例如，只要硬币不是匀质的，那么使用排列组合去计算就极有可能与观测到的实验情况不符——这是不科学的，也是不能被我们接受的。

概率的统计定义通常被认为是更为科学的。

在相同的条件下，随机试验 n 次，某事件 A 出现 m 次（ $m \le n$ ），则比值 $\frac{m}{n}$ 称为事件 A 发生的概率。随着 n 的增大，该频率围绕某一常数 p 上下波动，且波动的幅度逐渐减小，趋于稳定，这个频率的稳定值即为该事件的概率，记为：

$$P(A)=\frac{m}{n}=p$$

例如，有一枚匀质的硬币，一共扔出 1000 次，其中事件“正”为 498 次，事件“反”为 502 次。这个时候，下结论说“出现‘正’的概率为 49.8%”是可以的，说“出现‘正’的概率大约为 50%”也是可以的。这个过程我们同样可以用一段 Python 程序来模拟。

```
import random

coins = [0 for i in range(1000)]
proportion = [0 for i in range(1000)]

for i in range(1000):
  coins[i] = random.randrange(2)
  sum = 0
  for j in range(i+1):
    sum+=coins[j]
  proportion[i] = (float(sum)/(i+1))
  print '%.3f' % proportion[i]
```

最后输出的 proportion 中的每个元素 i，就是投掷到第 i 轮时出现硬币正面的比例值（如图 8–9 所示）。这个比例的图形最开始会抖动得很厉害，如果你做一次相同的模拟实验，也会和我一样看到剧烈的抖动过程，只不过具体的数值会不一样。而到最后，这个数值都会趋向于一个固定值——0.500，也就是 50.0%。

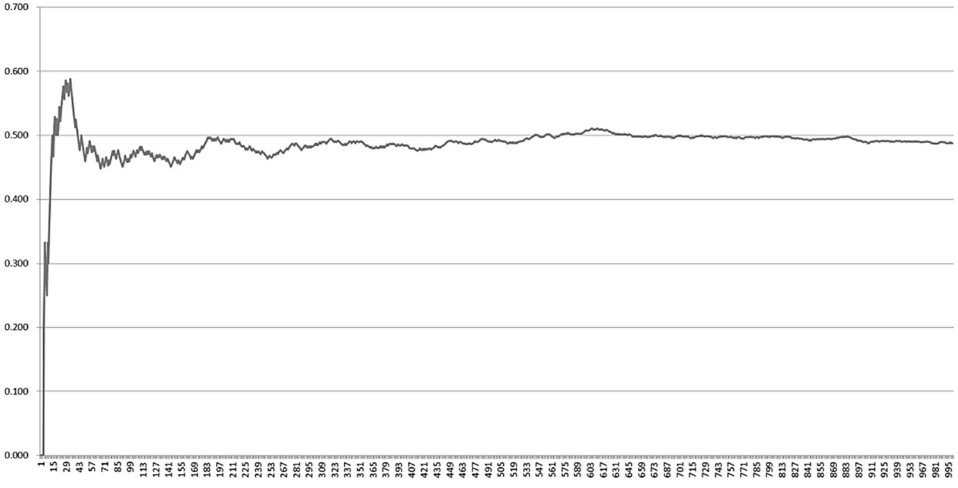


图 8–9　硬币投掷次数和统计概率

这种方式被认为更科学的原因是：它不是单纯地依赖一种“一厢情愿”的排列组合数量的推导，而是通过真实的实验得到数据，从获取方式上更接近科学的定义。要不怎么说“统计是最自然的认知方式”呢？

对于非匀质的硬币就不能使用这种方式进行模拟了，正确的方法是通过实际的观测来记录真实的实验结果。

主观概率往往是凭经验进行判断。这种方式的准确性，由于不是经过具体的实验得出的结论，所以不太容易作出科学的评价。虽然经验看上去也是一种“实验”，但是由于其准确性值得怀疑，所以在说服力方面也打了很大的折扣，在这里就不讨论了。

除此之外，概率还定义了加法法则，这样就使互斥事件的概率加和可以通过分别计算互斥事件发生的概率再做加和运算来得到。别看统计学是一种“高冷”的计算科学，但是它的应用性非常强，几乎在人类活动的各个领域都能起到认知和指导的作用。让我们往下看。

8.8.1　数学期望

在统计学中有一个非常重要的基础概念，叫作数学期望（Mean）。数学期望是指在试验中每次可能结果的概率乘以其结果的总和的平均值。

设在一次试验中，每一次的观测结果数量是一个随机变量 x_i，某一个 x_i 对应出现的概率为

$p(x_i)$，则其数学期望值为：

$$E(X)=\sum_{i=1}^{n}x_i p(x_i)$$

不难看出，这是一种加权平均的概念，也就是每次观测到的随机变量值加和再除以观测次数所得到的熵值。这是随机变量为离散值的情况。如果随机变量是连续值，可以写成积分的形式。设连续随机变量 X 概率密度为 $f(x)$，则数学期望值为：

$$E(X)=\int_{-\infty}^{\infty}xf(x)dx$$

这两种方式只是形式不同，其本质内容是一样的，都是加权平均的概念，即一次试验中观测到的随机变量平均值的概念，也是在描述一个随机过程观测值的时候最容易被人接受的描述方式。数学期望对描述一个大量或长期反复过程中的单次平均效果是有帮助的，也是很多基于统计的改进方案中一个重要的比对指标。

8.8.2　正态分布

正态分布应该是自然界中存在最为普遍的分布形态。我们在天文学、地理学、生物学、经济学、社会学等各个自然基础学科及社会学科的研究中经常能够观察到，很多统计样本的分布都呈现正态分布的特点。

正态分布的公式相信学过高等数学的读者都不会陌生：

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

正态分布也叫作高斯分布，是以德国著名数学家高斯的名字命名的，借此表彰他对这一数学理论的贡献（如图 8–10 所示）。

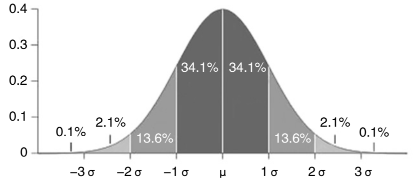


图 8–10　正态分布概率密度函数图

在统计学中，正态分布的相关应用是非常广的，我们观察到的很多事物都呈现正态分布。例如，在一定范围内随机抽取一定数量的人量取他们的身高，他们的身高就会呈现出正态分布的特点，画出的概率密度函数曲线是一个以身高平均值 μ 为中心，以 σ 为方差的钟形曲线。再如，在工厂制作的大量同等规格的零件中，抽取一定数量的零件精确量取其大小（例如长度或者直径等尺寸指标），通常也是一个以 μ 为平均值，以 σ 为方差的正态分布。在自然界中这样的例子还有很多，根本不用人为去做任何干涉，在统计上大量统计序列就会呈现出正态分布的样态。正是因为有了这样一种特性，在进行统计测量的过程中，为了减小误差，我们也会采用多次测量取平均值的方法来尽可能减小单次观测中引入的噪声。例如，评价当前社会个人财富水平，无论

这个社会体现出多么不平均的状态，也仍旧希望通过随机抽取一些人作为样本，然后取收入（财富）的平均值来进行衡量。在物理学上，如果希望相对精确地量取一个物体的长度，也要通过精确的量具进行多次测量再求平均值，并认为这个平均值是距离“真实值”最近的值。这一系列的理论依据仍旧是正态分布的理论依据，人们希望通过这种方式得到尽可能精确的 μ 值。可是，为什么它们会不约而同地呈现正态分布的特点呢？我们还是从刚刚这种统计和观察的过程说起。

因为万事万物都在不停地变化，所我们在观察任何一个对象的时候，都没有办法观察到一个“静态的”或“不变的”值。我们千万不能理解为：看到这个东西是“静止”的，它就一定是“静止”的。我们不能让这种错误的观念在拥有科学思维的头脑中蔓延。大到恒星、行星等天体，小到原子、电子等粒子，它们每时每刻都在运动。它们每时每刻都有自己运动的方向和速度，每时每刻都有自己具体的空间位置。问题是，人类的观测能力对它们来说永远都是有限的，在我们进行观测的一瞬间，多么希望有一个极其完美的“快照”状态把它们都记录下来——其实根本做不到。

人们在观察原子周围的电子时发现，电子也是一种人类无法用有效的方法描述其运动状态的微粒物质。所以，人们使用“电子云”模型的概念来描述它。以氢原子为例，简单地说，就是在任一时刻对原子进行一次“拍照”，把它的位置用一个点记录下来，然后在另一时刻同样进行一次“拍照”并做记录。持续进行这样的操作，当收集到足够多的“照片”后进行图像的叠加，就会发现大量的点集中在距离原子核比较近的位置，而在距离原子核相对较远的位置点比较少。这种由点的稀疏和稠密而产生的类似云状的概率描述模型叫作“电子云”（如图 8–11 所示）。不仅是氢原子，其他原子同样会产生类似的电子云效果。这种电子云本身给原子半径的测量带来了麻烦吗？这个麻烦究竟有多大？是应该计算电子云最外侧的圆形的半径，还是应该按照这些点的半径做加权平均？这些点的位置本身不就是在不停地变化吗？由这些不断变化的小粒子组成的物质岂不是在一刻不停地受到来自底层变化的影响？我们还有可能准确测得这些物质的尺寸吗？还是有科学的办法。我们要感谢棣莫弗和拉普拉斯，以及由他们发现的棣莫弗–拉普拉斯中心极限定理（如图 8–12 所示）。

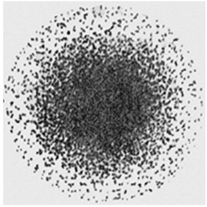


图 8–11　氢原子的电子云模型示意图

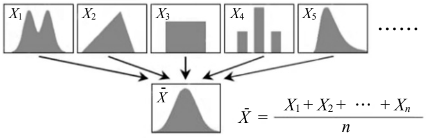


图 8–12　中心极限定理

设随机变量 $X_n(n=1,2,\cdots\cdots)$ 服从参数为 p 的二项分布，则对任意的 x 恒有

$$\lim_{n\to\infty} P\left\{ \frac{X_n-np}{\sqrt{np(1-p)}} \le x \right\} = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-t^2}{2}} dt$$

这种形态看上去已经有点像正态分布了。在一个二项分布中，只要样本足够多，有一个概率的概率密度遵循 $N(0,\ 1)$——μ 为 0，σ 为 1 的正态分布——这个概率就是 $\frac{X_n-np}{\sqrt{np(1-p)}} \le x$ 的概率。也就是说，只要 n 足够大，这个遵循二项分布（伯努利分布）的 X_n 的每一项与它的数学期望的差值，都是在 0 附近分布的情况多，向两侧延伸出去的情况少，服从正态分布。

其实不光是伯努利分布，其他任何分布的 X_n 都满足这一条件——当样本足够多的时候，任何随机变量 X_n 与它的数学期望的差值都服从正态分布。当然，不同的 X_n 与它的数学期望的差值所产生的正态分布的 σ 是不同的，而 μ 都为 0。不仅如此，如果 X_n 和 Y_n 独立且满足 $N(\mu,\sigma^2)$，那么令 $Z_n=aX_n+bY_n$，且 a、b 都为实数，Z_n 仍旧服从正态分布。

这种结论得出后，刚才我们谈到的物体长度测量问题就有理论依据了。即使每个原子的大小都是一个随机值，而在大量原子直径叠加的过程中，本应依靠测量大量原子的平均长度再加和，但由于这个测量的误差值是一个正态分布，所以就可以等价于测量大量原子加和后的总长度，这个总长度仍然会有一个误差值，而且这个误差值也仍然是一个正态分布，且永远无法消灭。除了原子自身大小的变化，其他任何本身误差都属于正态分布，而且可以由线性叠加施加给被测对象的值因素都会最终在被测量的值上以正态分布的误差表现出来，例如引力、磁场等。这些从组成世界的微小因子上体现出来的正态分布特性逐渐、逐层叠加，最终形成了现在体现在不同领域的各种各样的呈现正态分布的观测结果。

还有一个有趣的特性。假设 X_n 和 Y_n 是独立随机变量，$Z_n=aX_n+bY_n$，如果 Z_n 是正态分布的，那么 X_n 和 Y_n 也是正态分布的，这也称为“Cramer 分解定理”。这种方式为我们将大的研究对象分解为小的研究对象并对其采用同一种观测和分析方法提供了理论依据。这种正态分布可以叠加也可以分解的过程是一种非常“优美”的过程，它让我们可以使用同态的观测方式对形形色色的物质进行观测，并以此为依据进行误差减小的估算。

在我的理解中，正态分布是一种从宏观上体现出来的个体值与真实值差距的表示。正态分布线性叠加的结果还是正态分布，组成正态分布的线性叠加的因子本身也是正态分布，这种自相似式的分布特点，让众多数学家无比着迷。这是一种“超然存在”的状态，不管我们是否能意识到它，它一直这样存在着。我们能感知它的存在，但是无法改变它——这就是数学最大的魅力吧！难怪很多人会说，数学是离神祇最近的一种语言。这种说法或许并不夸张，因为无论如何，正态分布都可以说是所有分布中最为基础且适用范围最广的。

8.8.3 其他分布

除了正态分布以外，在统计学中还有很多分布特性可以用于解决一部分具有某些特殊性的统计问题，例如伯努利分布、泊松分布、伽马分布、卡方分布、T 分布、F 分布等。这些分布的适用场景都比较固定，也就形成了在相应场景中固定的使用方法和公式，成为定势性质的工具。

1. 伯努利分布

伯努利分布是专门研究一个随机过程中事件 X 发生的概率 p 与事件 X 不发生的概率 $1-p$ 的随机过程中的概率定量计算问题，表示为

$$P_n = \begin{cases} p & ,n=1 \\ 1-p & ,n=0 \end{cases}$$

满足伯努利分布的随机变量满足一个性质：可以通过这样一个公式来计算在 n 次试验中发生 k 次 $n=1$ 事件的概率。

$$P(X=k) = C_n^k \cdot p^k (1-p)^{n-k}$$

2. 泊松分布

泊松分布的参数 λ 是单位时间（或单位面积）内随机事件的平均发生率。泊松分布适合描述单位时间内随机事件发生的次数。在一个时间段内一个事件发生的次数为 λ 次，则发生 k 次的概率用下面这个公式来计算。

$$P(X=k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

3. 卡方分布

卡方分布（χ^2 分布）研究的是 n 个服从标准正态分布的随机变量 $x_1, x_2, \cdots, x_n$ 的平方和

$$Q = \sum_{i=1}^{n} x_i^2$$

构成的随机变量所遵循的分布规律。假设有 k 个独立标配正态分布，即 $n=k$（自由度）的概率密度公式为

$$f(k,x) = \begin{cases} \dfrac{1}{2^{\frac{k}{2}}\Gamma(\frac{k}{2})} e^{-\frac{x}{2}} x^{\frac{k}{2}-1} & ,x>0 \\ 0 & ,x \le 0 \end{cases}$$ [68]

68 伽马函数 $\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t} dt$ 。

当自由度趋向无穷的时候，χ^2 分布的概率密度函数仍然是一个标准的正态分布（如图 8–13 所示）。和其他分布概率密度的含义一样，当给定一个 x 值的时候，x 左侧的面积表示取值为 x 以下的值所占全部样本空间的比例。为了便于计算，在计算 χ^2 分布的概率值时都是采用查表的方式，即查阅 χ^2 分布表（或称“卡方分布临界值表”，如图 8–14 所示）。

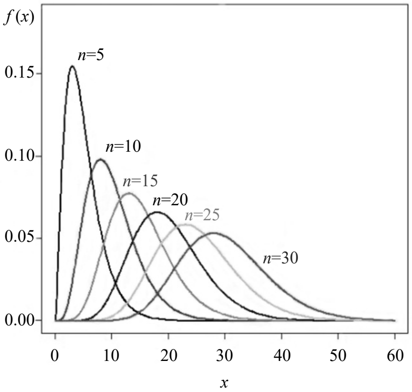


图 8–13　χ^2 分布概率密度

P / 自由度	0.995	0.99	0.975	0.95	0.05	0.025	0.01	0.005
1	0.000039	0.0002	0.0010	0.0039	3.8415	5.0239	6.6349	7.8794
2	0.0100	0.0201	0.0506	0.1026	5.9915	7.3778	9.2104	10.5965
3	0.0717	0.1148	0.2158	0.3518	7.8147	9.3484	11.3449	12.8381
4	0.2070	0.2971	0.4844	0.7107	9.4877	11.1433	13.2767	14.8602
5	0.4118	0.5543	0.8312	1.1455	11.0705	12.8325	15.0863	16.7496
6	0.6757	0.8721	1.2373	1.6354	12.5916	14.4494	16.8119	18.5475
7	0.9893	1.2390	1.6899	2.1673	14.0671	16.0128	18.4753	20.2777
8	1.3444	1.6465	2.1797	2.7326	15.5073	17.5345	20.0902	21.9549
9	1.7349	2.0879	2.7004	3.3251	16.9190	19.0228	21.6660	23.5893
10	2.1558	2.5582	3.2470	3.9403	18.3070	20.4832	23.2093	25.1881

图 8–14　χ^2 分布表

χ^2 分布通常用来度量一个分布的期望分布状态与多次抽样产生的分布的差异，并计算观测到的分布与期望分布有差异的概率是多少，这时只需要关注自由度和概率。例如，在观测中多次抽样，发现某种待求的概率多次浮动，记为 $P_1, P_2,\cdots, P_n$，那么得到

$$\chi_k^2(x) = \sum_{i=1}^{k} \frac{(P_i - P_e)^2}{P_e}$$

P_e 是期望概率。例如，自由度 k=10，选择显著水平 $p=0.005$。这个显著水平就是指置信区间，即置信区间为 99.95%，通过查表可以知道卡方值为 25.1881。它的含义是：如果 $\chi_k^2(x)$ 大于 25.1881，则随机变量 $x_1, x_2,\cdots, x_n$ 是正态分布的假设不成立。

8.9　统计学与大数据

从数理统计创立到现在，统计学已经有了 200 多年的历史。随着计算机处理能力的增强，通过抽样来推断整个样本空间数量的行为看似价值在降低。但是，放眼环顾，人类的认知水平永远无法企及自然界的事物总量——差距太悬殊！所以，这种通过抽样做推断的管中窥豹的认知理念丝毫没有过时，也从未过时。

一切测量，包括测量方法、误差分析等，都来源于最基本的对数据的观测和对认知的归纳。一切大量物质的叠加都遵从中心极限定理，在宏观上最终都会产生正态分布的积累效果。在正态分布的前提下，又有了一系列特殊性质的分布研究，例如卡方分布、F 分布、t 分布等，它们

都是非常好的用来度量和推测模型的工具。

在大数据产业蓬勃发展的今天，当我们意识到大数据会对社会产生的深远影响时，一定不要忘记：统计学在其中发挥着重要的认知引导作用。在后面有关数据分析的章节中，我们还能看到统计学的身影。

统计学能够帮助我们以简洁、高效、低成本的方式获取极高的认知准确度。还有什么比这更迷人？统计学是大数据存在的基础。

第 9 章　信息论

说到学习数据科学，信息论无论如何是绕不过去的。信息论是围绕数据存储与传输量化等一系列问题所展开的一门专门研究信息的学科，可以说是 20 世纪最伟大的理论发现之一，对之后的所有计算机数据压缩与传输科学的推动和发展都起着不可磨灭的作用。信息论的鼻祖就是大名鼎鼎的图克劳德・艾尔伍德・香农（Claude Elwood Shannon，1916 年 4 月 ~ 2001 年 2 月）。

信息论基础及其引论的内容非常多，不过归纳起来全部都是关于信息量的量化问题。如果非要用一句话来概括信息论研究的核心内容，那就是“在某种设定的情况下，最简洁可以用多少数据来表达或传输一个信息，同时量化它的失真程度”。这个问题被研究清楚以后，在计算机存储、压缩、通信传输甚至市场博弈中都有着很好的应用前景。

9.1　模拟信号

“信息是被消除的不确定性。”这是 1928 年由美国著名电子工程专家哈特莱[69]（如图 9-1 所示）提出的概念。

一段消息中包含的信息量究竟是多少？有多少是有用的、真正用来消除不确定性的信息？有多少不属于信息？乍听起来，这种研究似乎没什么特别了不起的价值，而仔细想想，研究清楚这件事情能够获得的收益是显而易见的。在我们周围每天都会产生数量极大的消息，不管是文字、音乐、图片、动态影像资料，还是其他任何形式的消息，如果我们希望将它们保存下来，就免不了要进行存储和传输。而存储和传输直接涉及实施难度——也就是成本的问题，甚至直接决定了方法是否可行。

图 9-1　哈特莱

在数字存储和数字通信出现之前，人们是通过模拟信号（Analog Signal）来进行数据存储和传输的。这种信号的记录和读取，与现在我们使用的 MP3 或者 WAV 文件格式极为不同——它们使用的是模拟信号。

69　哈特莱（R.V.L. Hartley），1888 年 11 月 ~ 1970 年 5 月。

以胶木唱片为例，其介质是树脂（如图 9–2 所示）。胶木唱片，又称“黑胶唱片”，是指转速 78 转/分，声槽宽度 0.10～0.16 毫米，声槽密度 30～50 条/厘米的留声机唱片。这种唱片是一种黑色圆盘形的胶片，用树脂（俗称“胶木”；后期改用聚氯乙烯，即 PVC）压制，上面刻有凹凸的坑纹以记录声音。

图 9–2　磁带和胶木唱片

声音是物体振动而发出的。无论是什么东西的振动，只要它们振动的各种属性（频率、振幅、波形等）完全一样，在我们听来就是一种声音。人耳能够听到的声音频率是有限的，这个范围通常被认为是 20～20000Hz。黑胶唱片是先把声音的振动属性记录在唱片上，就是在唱片上刻出一些弯弯曲曲的凹槽（“弯弯曲曲”就记录了音源的属性）。在回放声音的时候，需要把唱针放入凹槽中，当唱片按照一定的速率转动起来，唱片上的凹槽就会迫使唱针跟着振动，这样就有了声音。为了让人们能听清楚，需要通过放大电路将唱针的这种振动声还原成模拟电信号，然后将信号放大，从喇叭中播放出来（如图 9–3 所示）。

图 9–3　声音与模拟信号

这种介质的问题显而易见。第一是保存容易失真，受到磁头和唱针感应的影响，播放出来的声音会由于介质被侵袭而发生变化。第二是传输成本高，制作胶木唱片时，必须老老实实地刻录声音信息，然后通过人力进行传递。在老式的电话中，声音的传递是用同样的原理实现的，因此也就存在同样的问题——声音容易受到干扰，而且保密性极差。这些问题非常尖锐，亟待解决。

9.2　信息量与信息熵

在研究信息本质问题的过程中，哈特莱和香农作出了卓越的贡献，而他们研究的突破性也表现在这里。他们的研究没有局限于原先的各种以模拟信号组成的信道（包括介质），而是往前走了一步。信息究竟是什么？如哈特莱所说，“信息是被消除的不确定性”，那么消息的传输实际上就是要将这种不确定性变为确定性——第一步为什么不量化信息的大小呢？

1928 年，哈特莱提出了信息定量化的初步设想。他将在一次消息表达中所包含的不同含义数量 m 的对数定义为信息量。

$$I = \log_2 m$$

I 代表信息量的大小，m 用于表达不同含义的数量。

举个例子。在一次信息传递中，需要表达“是”、“否”2 个含义，只要送信和收信的双方对这个含义有共同的约定，那么不论这个消息是写在纸上，是通过电话传输，还是通过电报或其他方式传输，其信息量都是 $\log_2 2$，即 1。同理，要表达“上”、“下”、“左”、“右”4 个含义，则消息的信息量为 $\log_2 4$，即 2。

在信息量的计算过程中，对数 log 的底取 2。其实，在一次计算中，也可以通篇使用其他数字做底。用 2 做底主要是对应二进制的介质，对信道的解释表达得比较直观。

在二进制的介质中，在一个计数单位上只能用 0 和 1 两个数字来表示。所以，如果约定了 0 表示“是”，1 表示“否”，那么至少需要 1 位二进制介质的量才能清晰完整地表达消息的定义。如果要表达“上”、“下”、“左”、“右”4 个含义，1 位就不够用了，必须使用至少 2 位来表达。例如，00 代表“上”，01 代表“下”，10 代表“左”，11 代表“右”。否则，4 个不同的含义是无法采用比 2 更少的位数来完整描述的。以此类推，如果消息有 1000 种不同的含义，那么至少需要 10 位二进制介质来表示整个消息[70]。我们可以使用这种方式来理解哈特莱所提出的信息量的概念——确实很直观，也很有说服力。

香农对信息量的研究则更为深入且贴近本质。

既然“信息是被消除的不确定性”，如果内容确定的消息进行了传递，这些内容中还包含信息吗？根据定义，应该不包含信息。那么，这种不确定性本身是不是也能够量化呢？香农的高明之处就在这里，他引入了一种描述信息杂乱或意外程度的方式——信息熵。

$$H(x) = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i)$$

其中，x 代表某一信源，$P(x_i)$ 代表 x_i 消息产生的概率。

这个公式通过换算可以知道，哈特莱提出的 $I = \log_2 m$ 实际上是消息产生概率均等情况下的信息熵的特例。对于一个信源，它产生的消息中如果有某些产生的概率极大，就说明这种消息包含的信息量少；反之，产生概率极小的消息所包含的信息量就大。这种理解带来的深远影响，在后文中会有很多地方有所体现。

9.3　香农公式

在通信过程中，信道里的信号由于干扰问题，所以永远都存在误传的概率，这给远程通信带来了困扰。香农通过研究解决了一个非常重要的问题，那就是即使在有一定量噪声的信道环境中，通过对编码的调整，仍然可以获得没有误传的信号，并留下了传世的“香农公式”。

$$C = B \cdot \log_2(1 + \frac{S}{N})$$

70　2^9 是 512，$\log_2 512 = 9$；2^{10} 是 1024，$\log_2 1024 = 10$。

其中，C 是信道容量；B 是码源速率的极限值，$B = 2H$，H 为信道带宽，单位是赫兹；S 是信号功率，单位是瓦特；N 是噪声功率，单位是瓦特。

这个公式定量地揭示了，在噪声功率是 N，信号功率是 S，带宽是 $2H$ 的情况下，最大信息传输速率 C 的计算方法。不要小看这个公式，它为后来制造满足国际通信标准的同轴电缆、以太网线、光纤，以及解决如何基于这些介质进行编码和纠错的问题，提供了坚实的理论基础。

以我们常用的 Wi-Fi 信号 802.11n 为例，所用的频带是 2.422 ~ 2.462GHz，共 40MHz 带宽。在信噪比较好的情况下，例如 4.2dB[71]，可以这样计算：

$$C = 2 \cdot 40M \cdot \log_2(1 + \sqrt[10]{10^{4.2}}) = 148.8Mbps$$

这是在 40MHz 带宽下 802.11n 协议的带宽上限值。

如果发生信噪比降低的情况，例如背景噪声比较大，实际的传输速度就无法达到这个值，这时通常要加入一些冗余信号来进行信道纠错。

9.4 数字信号

前面我们提到过，在有了数字信号以后，数据传输中的抗干扰能力和加密能力都变强了。

先说抗干扰能力。在原先的模拟信号传输中，如果信道中有信号干扰，就会直接作用于信号，这对接收端而言就变得很麻烦，因为信号经过干扰已经发生了变形，也就无法得知原来发送的正确信号是什么了。

虽然数字信号在信道底层是以和模拟信号“无差别”的方式传输的，但因为数字信号有一定的编码规则，所以可以在传输过程中通过冗余信息来纠错。这种方式以在使用 Modem 于电话线上传输数字信号的场景中尤为典型。高频的计算机数据信号经过调制后，以小于等于 56kbps 的速率通过载波带宽为 3 ~ 8kHz 的电话信道传输（如图 9-4 所示）。

图 9-4 使用 4P4C 的 56kbps 调制解调器

在这种传输中，如果存在干扰，的确有可能使余弦载波在传输的过程中失真，进而导致接收端的误判，即在信噪比较低的情况下把 1 判断成 0 或者把 0 判断成 1。为了让通信更为可靠，在保证信息传播不失真的情况下就需要加入一些冗余信息来纠错。例如，在传输一个 8bit 数字的时候，实际编码只占 7 位，留 1 位纠错。

10101010

71 信噪比=$10\lg\frac{S}{N}$。

例如，前 7 位是对一条包含 128 个不同含义的消息进行编码，最后 1 位作为冗余校验（如图 9–5 所示）。在发信端发出这条消息之前，对前 7 位进行异或计算（XOR），把计算结果填入最后 1 位作为校验值。接收端拿到整个信息后，先检验前 7 位的异或计算结果是不是与最后 1 位吻合。如果吻合，就认为传输正确；如果不吻合，就认为传输错误，并要求对方重传。刚刚说的这种方式在一个信道的传输错误率严格小于 12.5% 的情况下是可行的。不过也能看出，一旦同时传错 2 位，就无法发现传输中的错误了。但是，对信噪比来说，更差的情况也就是误传率更高的信道，这时可以通过再增加 1 位校验位的方式纠错。原理与前面讲到的类似，只不过有效的信息编码只剩下 6 位，也就是只能对 64 个有效的信息进行编码。这种现象在我们使用 Wi–Fi 上网的时候并不鲜见，表现为信号越弱的时候上网速度越慢，原因是大量的消息传输都用来做冗余信息了。

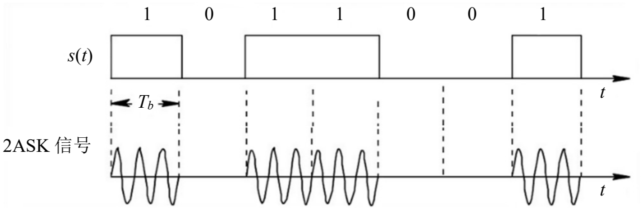


图 9–5　二进制振幅监控信号波形

这些对于编码的纠错和调整，都已经由通信领域的科学家们研究清楚，并将功能植入上网的模块中了，所以这类模块才能以极低的价格量产，并在实际应用中适应各种恶劣的环境。这是数据科学在通信和信息领域的一种完美体现。

9.5　编码与压缩

有了信息熵公式以后，信息学的发展就变得与原来大为不同。为了提高存储和传输的效率，越来越多的科学家致力于研究通过高效的编码算法和压缩算法来保证在不丢失信息的情况下使占用的空间或信道最少的问题。

早在 20 世纪 50 年代，美国的信息学教授戴维·哈夫曼[72]（如图 9–6 所示）还在麻省理工学院读博士的时候，在一篇名为 *A Method for the Construction of Minimum-Redundancy Codes* 的论文中就提出了一种编码方式，这就是著名的哈夫曼编码（Huffman Coding）。

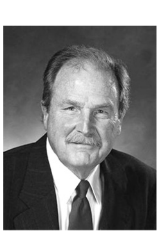


图 9–6　戴维·哈夫曼

在哈特莱提出的 $I=\log_2 m$ 的信息量计算公式中，通过信源种类 m 的数量可以得出使用多少位介质来表示一个信息。以 1 字节的二进制

72　戴维·哈夫曼（David Huffman，1925 年 8 月 ~ 1999 年 10 月），美国加利福尼亚大学计算机系著名教授。

数据为例，它有 8bit，最多可以表示 m 为 256 的情况。如果要传输 1000 字节，则需要在信道上传输 1000 组诸如 01010101、10100111、11000110 这样的 8 位编码，共 8000bit。

而根据香农的信息熵公式

$$H(x)=-\sum_{i=1}^{n}P(x_i)\log_2 P(x_i)$$

可以很快得出一个推论，那就是：只要每种信源产生的概率不均等，那么在刚刚这种场景中，传输 1000 组编码理论上肯定可以使用少于 8000bit 的数据，方法就是用比较短的编码来表示高频出现的字符。

我们以传输 6 种不同的字符信息为例（如表 9–1 所示）。

表 9–1　6 种不同的信源字符

	A	B	C	D	E	F
出现概率	50%	15%	10%	10%	10%	5%
定长编码	000	001	010	011	100	101
变长编码	0	10	110	1110	11110	111110

如果要传输 1000 组 3bit 定长编码信息，那就是需要传输 3000bit 的数据。但是，如果按照出现的概率不同，把大概率的数字换成较短的编码，情况会有所不同。

$$\begin{aligned}&L=1\times1000\times50\%+2\times1000\times15\%\\&+3\times1000\times10\%+4\times1000\times10\%\\&+5\times1000\times10\%+6\times1000\times5\%\\&L=2300\end{aligned}$$

在以上场景中，通过更换编码，可以把原先需要用 3000bit 传完信息用 2300bit 传完，起到了一定的压缩作用。

哈夫曼编码要保证一个叫作“前缀编码”的原则，即任何一个编码不能是另一个编码的前缀。原因在于，这种方式在接收到编码后进行解码时，由于无法判断当前接收到的编码是否为一个信息的结束标志，所以会产生困惑。

由于消息中产生的冗余信息会导致存储和传输效率低下，所以涌现出了一大批着力解决这方面问题的优秀算法。根据算法压缩时所涉及的场景，可以分为两大类，一类是无损压缩，另一类是有损压缩。

9.5.1　无损压缩

刚刚提到的对哈夫曼编码的应用就是无损压缩的一个最简单的雏形范例。无损压缩用于在压缩过程中不允许出现信息丢失的情况。例如，个人办公电脑中的各种客户资料，以及服务器

中的日志、交易信息等，无论以什么理由、使用什么压缩算法对其进行压缩，都不能以丢失信息为代价。

一些无损压缩基于字典压缩原理（Dictionary Compression）。字典压缩的原理和刚刚说的哈夫曼编码的思路类似，只不过它所统计的对象通常不是以单一字符为基础，而是以整个“单词”为基础。这里的“单词”是一个广义的概念，可以理解为连续出现的字符串。字典压缩的算法很多，例如 1977 年出现的 LZ77 算法、1993 年出现的 DEFLATE 算法及 1996 年出现的 LZO 算法等，到现在的应用仍然非常广泛。

以 LZ77 算法为例，围绕这个“字典”应该如何建立的问题，算法的作者下了一番苦功，要找出一段字符串中的重复结构信息，并根据它们来动态规划这个字典的建立规则。

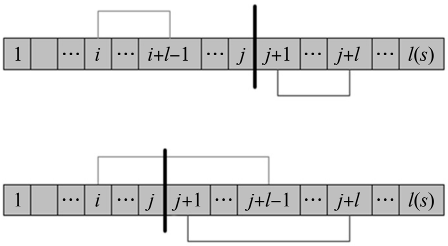


图 9–7　LZ77 算法的字符串匹配过程

LZ77 算法使用“滑动窗口”的方法来寻找文件中的相同部分，即匹配串（如图 9–7 所示）。这个串理论上可以是任意的串，并不局限于文字或文本信息。

LZ77 算法从文件的起始处开始，一个字节一个字节地向后处理。定义整个字符串（文件）为 S，长度为 $l(S)$。其中，$S(1,j)$ 为 S 的前缀，并且$1 \le j \le l(S)$。对于 $S(1,j)$ 且$i \le j$，$L(i)$ 为满足$l \le l(S)-j$的最大值，那么$S(i,i+l-1)=S(j+1,j+l)$。这样，字符串$S(j+1,j+l)$就和$S(i,i+l-1)$匹配了，匹配长度为 l。此时，定义一个 p 来记录最长匹配时的 i 值，并满足$L(p)=\max\limits_{1\le i\le j}\{L(i)\}$。

在这个计算模型中，一个固定大小的窗口随着处理的字节不断向后滑动。对于文件中的每个字节，用当前处理字节开始的串和窗口中的每个串进行匹配，以寻找最长的匹配串。窗口中的每个串是指窗口中以每个字节开始的子串。如果当前处理字节开始的串在窗口中有匹配串，就用

(之间的距离, 匹配长度)

这样一对信息来替换当前串。然后，从刚才处理完的串之后的那个字节开始继续处理。如果当前处理字节开始的串在窗口中没有匹配串，就不做改动地输出当前处理字节。

处理文件中第 1 个字节时，窗口在当前处理字节之前（也就是还没有滑到文件上时）没有任何内容，被处理的字节就会不做改动地输出。随着处理的不断向后，窗口滑入文件的部分越来越多。最后，整个窗口滑入文件，并在文件上向后滑动，直到文件结束。

为了在解压缩时可以区分“没有匹配的字节”和“(之间的距离, 匹配长度) 对”，我们还需要在每个“没有匹配的字节”或者“(之间的距离, 匹配长度) 对”之前，放上 1 位来避免混乱。例如，用 0 表示“没有匹配的字节”，用 1 表示“(之间的距离, 匹配长度) 对”。

在实际使用中，需要固定“(之间的距离, 匹配长度) 对”中的“之间的距离”和“匹配长度”所使用的位数。因为要固定“之间的距离”所使用的位数，所以采用了固定大小的窗口。例如，窗口的大小为 32KB，那么用 15 位（$2^{15}=32K$）就可以保存 0 ~ 32K 范围内的任何一个值。另外，需要限定最大的匹配长度，这样“匹配长度”所使用的位数也就固定了。

此外，在实际使用中，我们还将设定一个最小匹配长度，只有当两个串的匹配长度大于最小匹配长度时，我们才认为这是一个匹配。例如，“距离”使用 15 位，“长度”使用 8 位，那么“(之间的距离, 匹配长度) 对”将使用 23 位，也就是差 1 位 3 字节。如果匹配长度小于 3 字节，用“(之间的距离, 匹配长度) 对”进行替换的话，不但没有压缩，长度反而会增大，所以需要设置一个最小匹配长度。

解压缩的过程是这样的：从文件开始到文件结束，每次先读 1 位标志位，通过这个标志位来判断下面是一个“(之间的距离, 匹配长度) 对”，还是一个没有改动的字节。如果是一个“(之间的距离, 匹配长度) 对”，就读出固定位数的“(之间的距离, 匹配长度) 对”，然后根据对中的信息将匹配串输出到当前位置。如果是一个没有改动的字节，就读出 1 字节，然后输出该字节。

可以看到，LZ77 算法在压缩时需要做大量的匹配工作，而在解压缩时需要做的工作很少，也就是说，解压缩相对于压缩的速度要快得多。这对需要进行一次压缩、多次解压缩的场合是一个巨大的优势。而且，重复的内容越多，重复的字串越长，LZ77 算法的压缩效果就越好——这也是不言而喻的。

虽然不是每个人都有机会去编写 LZ77 这样的压缩算法，但每个服务器开发人员每天都会和它“亲密接触”。LZ77 算法是我们在 Linux 系统中使用的 GZip 算法的蓝本，真可谓“润物细无声”。

如果要测算 LZ77 算法的压缩率，可以找一些体积较大的文件来做压缩测试，效果总体来说还是不错的（尤其是文本文件）。

我用手边的文本文件做了一个简单的测试。

```
[root@ann char-rnn-master]# find / -type f -size +1000k -name "*.txt"
/usr/lib/python2.7/site-packages/netaddr/eui/oui.txt
/usr/share/perl5/Unicode/Collate/allkeys.txt
/usr/share/libhangul/hanja/hanja.txt
/usr/share/hwdata/iab.txt
/usr/share/hwdata/oui.txt
/ml/rnn/char-rnn-master/data/tinyshakespeare.1/input.txt
```

首先，用 find 命令找到本机上大于 1000KB 的 .txt 文件，通过命令查看其中第 1 个文件。

```
[root@ann char-rnn-master]# ls -la /usr/lib/python2.7/site-packages/netaddr/eui/oui.txt
-rw-r--r--. 1 root root 2249223 Sep 22  2010 /usr/lib/python2.7/site-packages/netaddr/eui/oui.txt
```

文件大小为 2249223 字节，即 2.25MB 左右。

```
[root@ann char-rnn-master]# gzip -9 /usr/lib/python2.7/site-packages/netaddr/eui/oui.txt
```

使用 gzip 命令将其压缩，参数 –9 表示使用压缩率最小的原则。

```
[root@ann char-rnn-master]# ll /usr/lib/python2.7/site-packages/netaddr/eui/oui.txt.gz
total 1572
-rw-r--r--. 1 root root 673972 Sep 22  2010 oui.txt.gz
```

压缩完成后，文件为 637972 字节，压缩率约为 28.36%，效率还是很高的。这类使用统计方法来动态规划字典的算法，应用非常普遍。

9.5.2　有损压缩

比起无损压缩，有损压缩更是一种每个人每天都在被动接触的一大类算法。这种压缩中允许有一定损失的场景多见于模拟信号的处理。

人类通过自身的视觉、听觉等感知到的模拟信号多为连续不断的信号。例如，有一个在时域上连续的声音信号，在以数字信号的形式对其进行记录时，需要对每个时间段中波形的波幅进行描述。用这种以离散方式记录波形的波幅，使得取样时间越宽，记录内容就与原始内容相差越大；取样时间越窄，记录内容就与原始内容相差越小。但是，要想采用数字信号对这些信息进行记录，势必无法避开从连续的模拟信号到离散的数字信号的转换过程（如图 9–8 所示）。从无限和连续，到有限和离散，无论怎么做，损失在这个环节都是无法避免的。

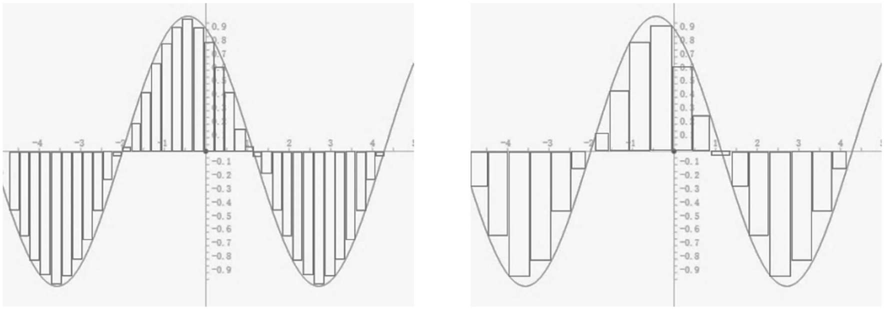


图 9–8　连续信号和离散信号

人类的视觉和听觉的敏感程度是有限的，也就是说，人类可以天然地容纳一些视听方面的信息损失，这也给多媒体信息的压缩带来了可能——只要使取样细密到一定程度，人类就不容易感知到其中的信息损失，所以，很多情况下也就没有必要去无限地追求这种取样的细密度。下面我们来认识一下这方面的应用。

1. 音频

WAV 文件格式是由微软研发的一种声音文件格式。标准格式的 WAV 文件和音乐 CD 使用的格式一样，都采用 44.1K 的采样频率，也就是在 1 秒之内进行 44100 次采样，最后生成的每一秒的声音信息都有 44100 个点的波幅来描述。这种采样频率已经非常高了，超出这个范围，人的耳朵就很难感知其中的变化了。

以人声为例，将这种声波的波形画出来，由于线条堆叠在一起，所以无法看出其形状（如图 9-9 所示）。但是，我们可以把它看成一个 $f(t)$ 函数，每一瞬间都是在 0 点上下往复抖动的周期为 60 ~ 4000Hz 的波幅不断变化的曲线。

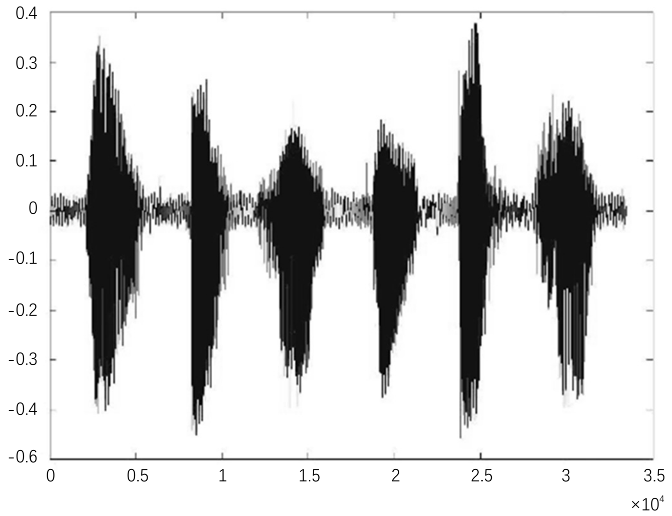


图 9-9　声波波形

振幅量化有 8 位、16 位、24 位 3 种精度，声道也有单声道和双声道之分。以高精度的 24 位双声道，采样周期为 44100 次/秒来计算，存储 1 秒的声音需要的存储空间为：

$$C = 16 \times 44100 \times 2 = 1411200\text{bit}$$

也就是每秒 176.4KB，每分钟 10.584MB。

如果使用无损压缩，则采用任何与普适的文件无损压缩没有差异的压缩算法就可以了。这里我们不多讲了，只说一下现在最常见的 MP3 格式究竟在压缩这个环节做了什么。

MP3 的全称是"MPEG Audio Layer 3"，它是一种高效的音频编码方案，以较大的压缩比将音频文件转换成较小的扩展名为 .MP3 的文件，并能基本保持原始文件的音质。

MP3 采用了感知音频编码（Perceptual Audio Coding）这一失真算法。人耳感受声音的频率范围是 20 ~ 20kHz，MP3 去掉了大量的冗余信号和无关信号，编码器通过混合滤波器组将原始

声音转换到频率域中，利用心理声学[73]模型估算刚好能被人耳察觉的噪声水平，再经过量化，将其转换成哈夫曼编码，形成 MP3 位流。解码器的工作就简单得多，主要是从编码后的谱线成分中通过反量化和逆变换提取声音信号。常见的 WAV 和 MP3 格式的频谱信息如图 9-10 和图 9-11 所示。

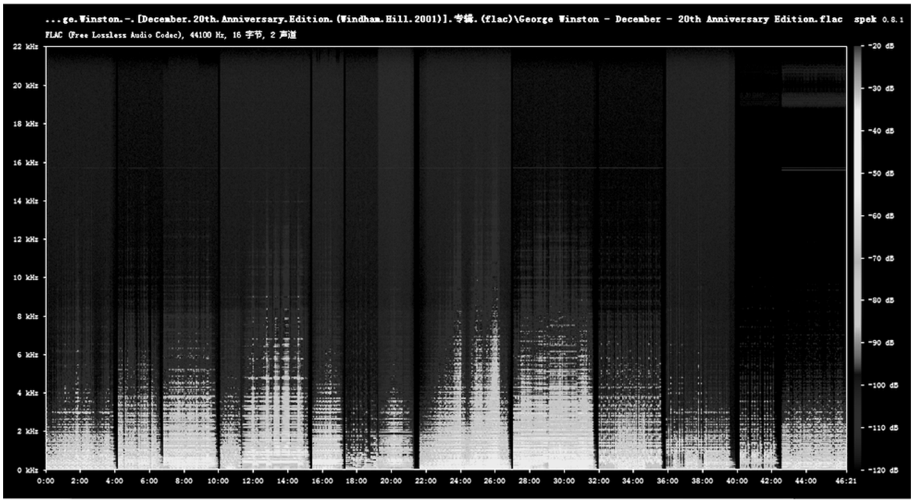


图 9-10　WAV 频谱信息

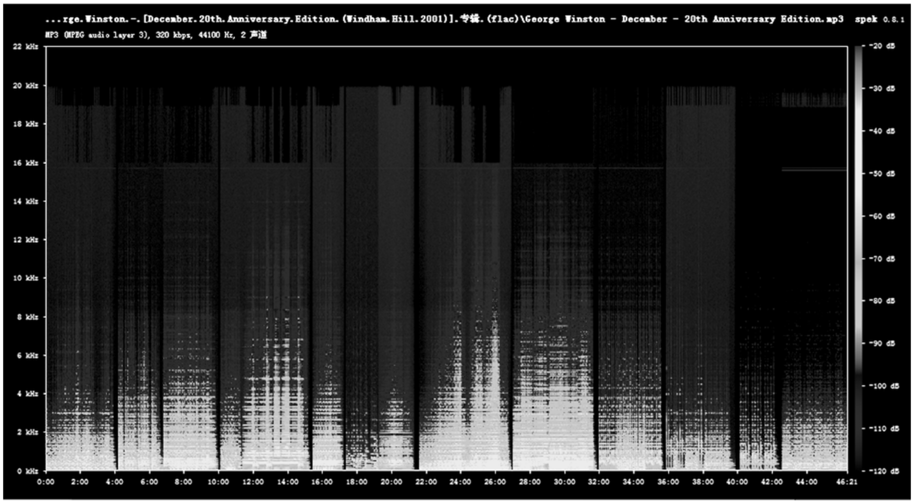


图 9-11　MP3 频谱信息

MP3 格式去掉了那些人耳不太敏感的高频分量，减少了记录的信息，所以才具有高音质和高压缩比的特性。

73　心理声学（Psychoacoustics）是研究声音和它所引起的听觉之间关系的一门边缘学科。

以中等码率 192kbps 的 MP3 文件来说，每秒音频占用 192kb 的空间（流量），而每分钟则占用 1.44MB（如表 9-2 所示）。

表 9-2　不同频率的采样标准及其用途

频　　率	用　　途
8kHz	电话使用
22.05kHz	广播使用
44.1kHz	音频 CD
48kHz	DVD、数字电视中使用
96 ~ 192kHz	DVD-Audio、蓝光高清等使用

在不同的应用场景中，采样率通常会有区别，而这也要视场景特性而定。对语音电话来说，只要保证沟通的内容能够听清楚就可以了，所以使用 8kHz 尽管音质不完美，却节约了大量的信道带宽资源。而在 DVD 蓝光高清这样的娱乐场景中，追求的是品质，所以在这种情况下会更注重音频的细腻性。这样的取舍逻辑我想每个人都会有一些感悟。

2. 图片

在我们每天使用的办公电脑中，通常有两大类静态图形文件，一类是矢量图，另一类是标量图，后者更为常见。矢量图和标量图的最大不同在于图中信息的描述方式，当然这也为它们带来了各自的优势和局限性。

（1）矢量图

矢量图一般有用 Corel Draw 打开的 .CDR 格式、用 Illustrator 打开的 .AI 格式、用 Photoshop 打开的 .EPS 格式等。矢量文件对图形中的每个元素对象进行描述，每个元素对象都是一个独立的实体，描述的信息包括颜色、形状、轮廓、大小和位置等属性，就连字体信息也都保存为一个字体的描述指针。3d Max 这种用来做三维模型的专用软件，其文件也都是以这种对象和模型的描述形式来保存的（如图 9-12 所示）。

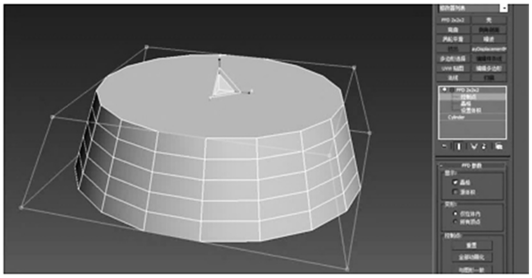


图 9-12　3d Max 中的矢量模型文件可视化效果

这种描述方式的优势在于，由于描述的是一个对象信息，那么这个描述可以用“连续”的方式来表达。例如，在一个平面中声明一个圆形的信息，需要用哪些参数来描述呢？

圆心坐标 (x, y)，圆的半径 r，圆的颜色 RGB。从理论上说，在 64 位计算机操作系统中，用 48 字节就能把这个圆的信息描述完成，对立方体则描述顶点位置与颜色即可。如果想将图形放大，也是没有限制的，放大过程中不会产生锯齿效果（看看后面关于标量图的介绍就能知道区别了）——这种描述是一种“连续型”的描述，可以“延展”而不失真。

矢量图当然也有局限性，那就是当文件中描述的图形比较细碎的时候，描述起来就比较困难（因为对每个对象都要声明其颜色、形状、轮廓、大小和位置等属性）。很明显，用矢量图描述简单的图标或图形对象信息还是比较方便的，而对于人眼能够识别的曲度、颜色等极为丰富的图形信息则变得极为不便。

（2）标量图

一般来说，人们在记录（例如摄影照片、大幅面不规则且色彩丰富的画面信息）的时候会使用标量图。比较早的标量图就是微软在 Windows 中使用的 .bmp 后缀的 BITMAP 格式。在局部放大以后，我们能明显看到其中有锯齿的效果，原因在于这种文件内容的描述方式是基于像素单位的离散描述（如图 9–13 和图 9–14 所示）。

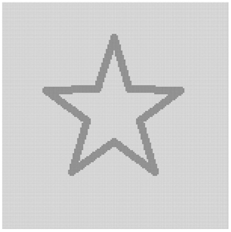


图 9–13　150×150 的位图文件

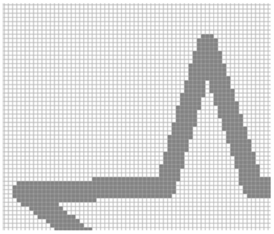


图 9–14　150×150 的位图文件的锯齿效果

这种格式的描述是：在一个平面直角坐标系中，以左上角为点 (0, 0)，向右为 x 轴正方向，向下为 y 轴正方向，每一个点都用一个 24 位的 RGB 通道来描述，那么这个 BMP 文件的大小应该是：

$$C = 3 \times 150 \times 150 = 67500$$

67500 字节，也就是 67.5KB。加上文件描述等额外开销，这个文件实际占用了 67854 字节。将它保存成 .JPG 格式以后，文件的占用空间明显变小了（如图 9–15 所示）。文件的大小缩小到 5401 字节，仅为原来的 7.96%，而像素数仍旧是 150×150，这一点没有变化（如图 9–16 所示）。既然我们已经有了无损压缩的思路，那么从这个文件的特点上也能看出一些端倪。大量的白色信息恰恰是一种字符串高频度冗余，所以使用字符串压缩的算法也能够对刚才这个 .BMP 文件进行无损压缩。不过，.JPG 格式是怎么做到的呢？应该说，这种有损压缩更为高明。

图 9–15 .BMP 文件占用的空间

图 9–16 .JPG 文件占用的空间

与 MP3 压缩的思路一样，JPEG[74] 在对数字图片进行压缩的过程中也对人类感知进行了研究，发现人眼对颜色中亮度变换的敏感度比对色彩变换的敏感度高很多，于是就尝试着在编码中用角度的数据空间对亮度进行描述。所以，JPEG 中对颜色的描述其实不是 RGB 三通道方式，而是 YCbCr 颜色空间方式，转换公式如下：

$$Y=0.299R+0.587G+0.114B$$
$$Cb=-0.1687R-0.3313G+0.5B+128$$
$$Cr=0.5R-0.418G-0.0813B+128$$

对应的逆运算也能得到：

$$R=Y+1.402(Cr-128)$$
$$G=Y-0.34414(Cb-128)-0.71414(Cr-128)$$
$$B=Y+1.772(Cb-128)$$

这里没有什么技巧，只是三元一次方程组的运算。下一步的采样比较重要。

目前主要的采样格式有 *YCbCr* 4∶2∶0、*YCbCr* 4∶2∶2、*YCbCr* 4∶1∶1 和 *YCbCr* 4∶4∶4，几种标准的名称分别是 YUV420、YUV422、YUV411、YUV444[75]，其中 YUV411 比较常用。

以 YUV411 为例，每个点保存一个 8 位的亮度值（也就是 Y 值），每 2×2 个点保存一个 Cb 和 Cr 值，图像在肉眼中的感觉不会有太大的变化。所以，理论上使用 RGB 模型每个点需要 24 位来描述，而使用 YUV411 仅需要 12 位。这种方法从抽样环节开始就进行了有损压缩，损失的部分是人眼相对不敏感的色彩变换信息。

下面就是进行 DCT 变换，即离散余弦变换（Discrete Cosine Transform）。DCT 是指将一组光

74 联合图像专家小组（Joint Photographic Exports Group），隶属 ISO 国际标准化组织。

75 YUV 是欧洲电视系统采用的一种颜色编码方法，Y 表示亮度（Luma），U 和 V 表示色度（Chroma）。

强描述数据转换成光谱频率描述数据。这个地方非常有意思。

$$b(x,y)=\cos[\frac{(2x+1)u\pi}{2N}]\cos[\frac{(2y+1)v\pi}{2N}]$$

其中，x 和 y 代表图形数据矩阵某点的坐标，u 和 v 是奇函数频率域中的坐标——相当于傅里叶变换中的 ω；N 是块大小，习惯上取 8，也就是在 8×8 的空间里进行离散余弦变换。这种思想实际上可以认为是把音频信号中的 $f(t)$ 迁移到二维离散空间来使用。变换完成后，每个波形利用无限多的不同周期的余弦波叠加而成的性质可以用来做周期的取舍，优先处理低频分量，通过舍弃高频分量来牺牲对于波形还原的精度。

这种变换完成之后，在一个 8×8 的矩阵中，有 64 个点的信息表示 64 个不同的波的振幅信息，低频分量的信息都集中在左上角的 (0, 0) 点，而向右向下，高频分量的信息增加，在 (7, 7) 点集中了所有的高频信息（如图 9–17 所示）。

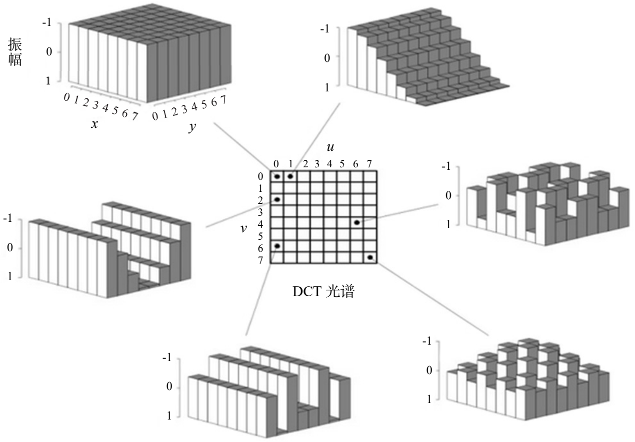


图 9–17　DCT 变换中的基函数

在这个环节可以通过对高频分量的舍弃来起到压缩的作用。如图 9–18 所示：图 d 表示的是 64 个频率描述信息叠加的波形效果，图 a 是 3 个，图 b 是 6 个，图 c 是 15 个。通过观察可以发现，在这个例子中，15 个频率分量信息叠加的结果和 64 个已经很接近了，而这仅仅用了不到四分之一的频率信息。

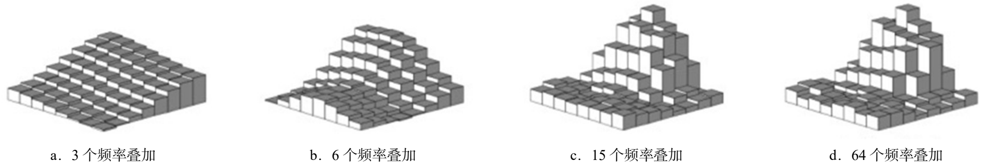


图 9–18　不同频率分量叠加的还原效果

在前面获得 8×8 的频率振幅描述矩阵之后，再用这个矩阵与量化表中相应的值相除来进行振幅的压缩。如图 9–19 所示：左上角的除数比较小，而右下角的除数比较大，这就意味在相除之后低频的描述信息仍然比较丰富，而高频信息几乎没有了；左侧的低压缩率量化表和右侧的高压缩率量化表只是商的系数不同，高压缩率量化表在做除法的时候，显然由于除数很大而得到了更大的压缩幅度。

最后就是进行遍历和对遍历后的序列进行无损的哈夫曼压缩编码（如图 9–20 所示）。

低压缩率

1	1	1	1	1	2	2	4
1	1	1	1	1	2	2	4
1	1	1	1	2	2	2	4
1	1	1	1	2	2	4	8
1	1	2	2	2	2	4	8
2	2	2	2	2	4	8	8
2	2	2	4	4	8	8	16
4	4	4	4	8	8	16	16

高压缩率

1	2	4	8	16	32	64	128
2	4	4	8	16	32	64	128
4	4	8	16	32	64	128	128
8	8	16	32	64	128	128	256
16	16	32	64	128	128	256	256
32	32	64	128	128	256	256	256
64	64	128	128	256	256	256	256
128	128	128	256	256	256	256	256

图 9–19 JPEG 量化表

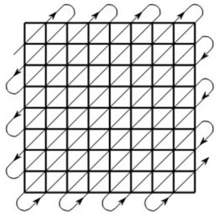


图 9–20 遍历与编码

对图像压缩信息进行解压缩是刚刚这些步骤的逆过程，因为在压缩中有一些步骤是有损的，对高频分量进行了舍弃，所以在还原的时候无法获得和原始的 RGB 三通道或 BITMAP 相同的精度，但这个问题造成的影响很有限。

今天在互联网中大量使用的 .JPG 文件就是基于这样的一种压缩思路，是不是非常巧妙呢？互联网能有如此之快的发展速度，与信息论及由它作为理论基础所发展而来的一系列强大的算法理论体系有着密不可分的关系。

3. 视频

在本节的最后，我们讨论一下视频信息的压缩问题。

视频是现在在互联网环境中非常常见的一种媒体形式，不论是视频流还是视频文件，我们天天都会接触。

视频是一种典型的多媒体信息。我们通常说的视频是指图像信息和音频信息两个独立部分叠加的内容。对老的国产动画片有研究的读者估计会比较清楚，不论是像《大闹天宫》这样的工笔画动画片，还是像《阿凡提》这样的银丝木偶动画片，都是一帧一帧拍出来的（如图 9–21 所示）。

人眼对连续运动的物体的敏感度也是有限的，一幅一幅的静态画面快速连续播放，就能让人得到视觉动感。一般来说，大于 30fps[76] 就不会让人感到明显的动作跳跃感，换言之，视频文件的画面部分存储了大量的一帧一帧的静态画面信息，然后快速播放给观众看，这种技术原理在视频流和视频文件中通用。在前面我们已经知道，单幅静态画面可以通过离散余弦变换来进

76 帧每秒（Frame Per Second），用来描述视频播放速度的技术指标。

行信息压缩（牺牲一些高频分量来获取较高的压缩比），那么对动态画面有什么更好的处理方法吗？也是有的。

图 9-21　动画片《大闹天宫》和《阿凡提》

以 H.264[77] 编码标准为例，由于视频本身的特点，相邻的帧之间通常相似度比较高，像素、亮度、色度的变化都非常小。所以，对一段变化不大图像，可以考虑用数据增量的方式实现，这样做会节省大量的传输带宽。

H.264 编码标准定义了 3 种帧（如图 9-22 所示）。完整编码的帧叫作 I 帧，又称“内部画面帧”（Intra Frame）。I 帧中是一帧完整的图像信息。参考之前的 I 帧生成的只包含差异部分编码的帧叫作 P 帧，又称“预测帧”（Predictive Frame）。还有一种参考前后的帧编码的帧叫作 B 帧（Bi-Directional Frame）。当把一帧压缩成 B 帧时，它会根据相邻的前一帧、本帧及后一帧数据的不同点来压缩本帧，也即仅记录本帧与前后帧的差值。只有采用 B 帧压缩才能达到 200：1 的高压缩比。H.264 编码标准采用的核心算法是帧内压缩和帧间压缩相结合的方法，帧内压缩是生成 I 帧的算法，帧间压缩是生成 B 帧和 P 帧的算法。

图 9-22　时间序列上的 I 帧、B 帧和 P 帧

一个 I 帧到下一个 I 帧之前的部分叫作一个画面组（Group of Pictures，GOP）。接收端收到连续不断的 GOP，就可以连续不断地根据 I 帧、P 帧和 B 帧中记录的差异量逐帧进行完整的复原，再在显示环节进行渲染，这样就能还原出原始的视频信息。在一个 GOP 中，只要 I 帧不丢失，那么即使有个别 P 帧或者 B 帧丢失，对画面感官的影响也是有限的。如果 I 帧丢失，那么整个 GOP 所附带的信息就没有还原的可能了，画面观感会非常差，有可能会严重花屏或黑屏，这样就只能等着下一个 I 帧的到来，从而得到一帧完整的画面信息。

如果读者做过 DBA，或者对 Oracle 数据库比较熟悉，就能很容易联想到，这种方式和 Oracle 进行全量备份和增量备份的方式相同。通过一次完整的库扫描，把数据块的信息用 RMAN 进行备份，之后要进行增量备份时，只需要备份那些从上次备份时起到当前时间为止的有变化的块信息，而不需要备份整个数据库，节省了大量的时间。

77　H.264 是国际标准化组织（ISO）和国际电信联盟（ITU）共同提出的继 MPEG4 之后的新一代数字视频压缩格式。

9.6　本章小结

在这一章中，我们讨论的是关于信息熵和信息量的问题，以及利用这种规律进行的数据压缩与传输的整个理论依据和技巧思路问题。

除了在有损压缩的部分利用人对模拟信号的不敏感性进行了信息丰富程度与压缩比的取舍以外，剩下的各种关于数据压缩的思路都是从信息论中关于信息和信息冗余的思路发展而来，真可谓殊途同归。

对信息论的讨论到此告一段落。如果读者对这部分内容有更大的兴趣，可以参阅其他技术论文和专业书籍。对纯粹的工程层面应用来说，只要牢记信息承载中这种由于信息出现的概率不同而产生的无损压缩应用就足够了。

第 10 章　混沌论

在英国有一首著名的民谣，名为 *For Want of a Nail*。

"For want of a nail the shoe was lost.
For want of a shoe the horse was lost.
For want of a horse the rider was lost.
For want of a rider the message was lost.
For want of a message the battle was lost.
For want of a battle the kingdom was lost.
And all for the want of a horseshoe nail."

中文大意为：

"丢失一颗钉子，坏了一只铁蹄；
坏了一只铁蹄，折了一批战马；
折了一批战马，伤了一位骑士；
上了一位骑士，输了一场战斗；
输了一场战争，亡了一个帝国。
这一切都是因为一颗马蹄钉。"

很遗憾，这首民谣最早的出处我没有找到，不过它确实有多个版本并且流传了几个世纪。其中，"For want of a nail" 这句谚语最早、最完整的出处是英格兰理查三世阵亡于博斯沃斯之战[78]的故事。在一本名为 *Fifty Famous People* 的寓言故事集中，绘声绘色地记载了理查三世的马夫催促铁匠给国王的战马钉马掌，但是钉子不够，为了赶时间只能硬着头皮让国王骑乘缺少马蹄钉的战马上阵拼杀，结果马掌掉了，战马摔倒，理查三世也掉落战马……最后，他被对手亨利战胜，亨利成为英格兰的国王。不过据史书记载，理查三世的战马在战斗中仅仅是陷在淤泥里而已。这些丰富的补充信息，大概是一代代艺术家为了增加其趣味性和意外性进行的演绎吧。

78　博斯沃斯之战发生于 1485 年，是兰开斯特王朝和约克王朝之间的战争中最重要的一场战役，导致了约克王朝最后一任国王理查三世的死亡，是英国历史上重定乾坤的重要战役。

在这个小故事里，我们能得到的最清晰的一种示意就是：一个小小的事件，很可能由于某些巧合而放大，在一段时间以后产生一个看上去很重大的结果。我相信不少读者可能会觉得这个故事中有不少夸张之处，而且会对少几颗钉子就能导致一个帝国覆灭这样的故事是不是在捕捉巧合而忽略了背后更重要的原因存疑，例如我们平时说的“人心向背”、“生产力水平高低”等大趋势方面的问题。

如何理解这些微小的“巧合”叠加所产生的“严重后果”呢？是否可以考虑用数学模型的方式去尝试去观察、量化或解释这些问题呢？这些“巧合”对未来的影响究竟有多大呢？这些恐怕才是混沌论这个学科最为关心的问题。

10.1 洛伦兹在想什么

美国著名的气象学家洛伦兹[79]（如图 10–1 所示）在 1963 年于麻省理工学院做长期天气预报研究的时候发现了一个现象。那段时间，他尝试对大气对流模型进行数值计算，并观察这个系统的演化过程。他将温度、湿度、压力等气象数据输入计算机，计算机会依据 3 个内建的微分方程计算出下一刻可能的气象数据，据此模拟绘制气象变化图。但是，在这个过程中发生了一件非常奇怪的事情：两次输入“相同”的数值，用曲线表示的计算结果竟然相差很远。刚开始，两次计算的结果吻合度还很高，但是慢慢地，两条曲线的距离越来越远，到最后甚至看不出两者有丝毫关系（如图 10–2 所示）。

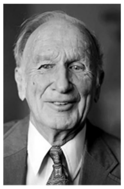


图 10–1 气象学家洛伦兹

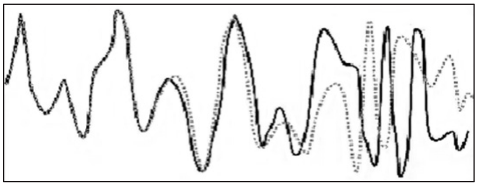


图 10–2 洛伦兹两次实验绘制的曲线不同

对计算结果进行分析后，洛伦兹找到了原因。原来他在第 1 次计算中输入了一个参数，值为 0.506127，而在第 2 次计算中输入的参数值为 0.506——差别仅在这个地方。

在中短期时间内，这个系统有着相对确定的规则行为，但是时间一长，在同一系统中会发生无规则的行为。换句话说，在大气对流模型建模的过程中，虽然考虑得很缜密，并且利用了电子计算机对数据进行快速、精确的计算，但仍旧无法在长时间周期里获得稳定的输出解——长期天气预报没有实现。在这个“系统”初始化的时刻，给定的参数即使只有很小的差别，也会在足够长的时间之后将系统的输出差距增加到没有任何联系的地步，而这个初始化的参数的精

79 气象学家洛伦兹（Lorenz Edward Norton，1917 年 5 月 ~ 2008 年 4 月），与物理学家亨德里克 · 安东 · 洛伦兹（Hendrik Antoon Lorentz，1853 年 7 月 ~ 1928 年 2 月）不是同一人。

度不论多高，总会有误差存在，且一定会与理想的真实值有无法忽略的差。1963 年，洛伦兹发表了一篇题为《确定性非周期流》[80] 的论文，之后又连续发表了几篇著名的论文[81]，这些论文逐步形成了研究耗散系统混沌现象的经典文献。

著名的"蝴蝶效应"就是混沌现象为大众所知的一种气象学体现。概括地说，这个词汇的含义就是"亚马逊雨林一只蝴蝶翅膀偶尔振动，也许两周后就会引起美国德克萨斯州的一场龙卷风"。这种说法不是夸大其辞，尤其不是为了说明"一只蝴蝶翅膀偶尔震动"是"两周后美国德克萨斯州的一场龙卷风"的直接诱因，而是为了说明——某一微小因素在多轮叠加后会被放大，诱导产生一种与在这一微小因素不发生的情况下截然不同的结果。

真的会这样吗？是的。混沌理论整体研究的就是这样一类事情，而且进行的是定量研究。

10.2　罗伯特·梅的养鱼计划

罗伯特·梅[82]（如图 10–3 所示），1938 年出生于澳大利亚悉尼，科学家，涉猎很广，对理论生态学、人口动态、生物系统的复杂性和稳定性等问题有深入的研究，1973 年之后研究完全转向生物学领域。

图 10–3　罗伯特·梅

在对生物种群的个体数量进行研究的过程中，人们最初使用这样一种线性函数进行描述：

$$x_n = kx_{n-1}$$

这里的 n 和 n–1 表示相邻的两个观察周期或者生物代。这种思路就是：假设任意的生物种群，无论是大象、长颈鹿，还是鱼、虾、飞鸟等，都有一个自然的增长率。不过，这种假设很快就被人发现有着极大的局限性，因为这种增长可以说是一种无限制的增长。虽然这种模型假说附和了马尔萨斯有关"人口增长不受食物供给和道德约束限制"的"封口增长"理论，但这和我们观察到的现象大相径庭。

罗伯特·梅及其前人在研究中发现，用一种变形后的公式来描述比较符合生物界的常识。

$$x_n = kx_{n-1}(1-x_{n-1})$$

这个模型弥补了线性模型假设中的瑕疵：当种群数中的个体数量较小的时候，食物供应量非常充足，种群中的个体数量就会增加；当种群中的个体数量非常大的时候，食物供应量会不足，种群中的个体数量就会减少。

80　1963，*Deterministic nonperiodic flow*，*Journal of Atmospheric Sciences*。

81　1967，*The nature and theory of the general circulation of atmosphere*，*World Meteorological Organization*。1969，*Three approaches to atmospheric predictability*，*Bulletin of the American Meteorological Society*。1972，*Predictability: Does the Flap of a Butterfly's Wings in Brazil Set Off a Tornado in Texas?* 1976，*Nondeterministic theories of climate change*，*Quaternary Research*。

82　罗伯特·梅（Robert May，生于 1938 年），牛津大学教授，英国皇家学会会员。

这种设计看上去确实很完美，不过遵循这种模型的研究很快就遇到了麻烦——罗伯特·梅在研究鱼类种群个体数量的时候碰到了问题（如图 10–4 所示）。

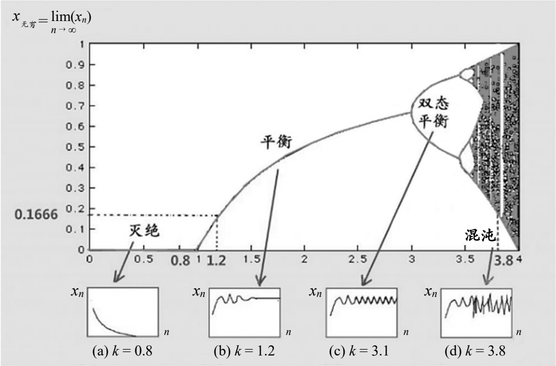


图 10–4　鱼群个体数量混沌态示意图

从图 10–4 中可以清晰地看到：当 $k=0.8$ 或者取更小的值时会发生衰减现象，不管初始状态的 x_1 是多少，种群最终都会消亡；当 k 的值介于 1 到 3 之间时，会出现平衡态的效果，即种群中的个体数量会趋于一个定值；当 k 介于 3 到 3.5 之间时，会出现“双态平衡”的现象，也就是说，最终的种群中的个体数量会由于 x_1 的不同而收敛到两个不同的值上去；当 k 取更大的值（例如 $k=3.8$）时，则会出现完全不确定的收敛值，最终种群中的个体数量完全无法预测。

这种现象就是生物种群个体数量研究领域出现的混沌现象。感兴趣的读者可以用 Excel 来模拟这个过程，你会观测到和罗伯特·梅一样的现象。

为什么会产生这种现象？在其他领域也会产生这种现象吗？我们很快就会知道答案。

10.3　有限的大脑，无限的维

在人类通过数学建模研究客观世界的过程中，任何一个时代都有才华横溢的科学家向世人展示着他们的技巧。

不管是在古希腊时期还是在今天，这些科学家在建模的时候无一例外，都要不断对观测对象进行测量和归纳，将观测到的数值进行整理，并用一系列的公式描述它们之间的关系。世界是怎么被创造出来的并不重要，反正人类只能根据当前的认知能力来进行归纳和推测。对于人类有限的大脑来说，能做的只是让自己描述的模型和观察到的现象的差距尽可能小。真正无穷精确的模型是永远不可能得到的。

对于“不变”的数值，研究起来相对比较容易。例如，《几何原本》中研究的都是固定的几何形状的量的关系，任何因素在其中都没有迭代的影响，所以当一部分因素确定后，其他因素同样会确定下来。对于变化的量，描述方式就相对复杂一些。在 3.2.3 节我们已经介绍了在连续

变化的模型中使用微分方程组进行描述并求解的实例，而在一些观测结果呈现离散特性的模型中，会使用基于数列前后关系描述的模型。

$$a_n = f(a_{n-1})$$

这种离散的形式也很常见，等比数列、等差数列等都属于其典型方式。

如果它们呈现

$$a_n = ra_{n-1} + b$$

这种关系，那就是线性动力系统。

如果其中产生了二次项甚至更高次的项，那就是非线性动力系统，例如

$$a_n = r(1-a_{n-1})a_{n-1}$$

这种建模方式同样是基于大量观察样本特性并尝试归纳其关系的过程得到的，而且这样的归纳方式也可以用于求解多个序列之间的数值关系描述和，例如

$$a_n = ra_{n-1} + sb_{n-1}$$
$$b_n = ka_{n-1}$$

在系数确定的情况下，一旦给定 a_1 和 b_1 的值，就可以得到整个数列。无论 n 的值多大（时间过了多久），都能通过这样的方式“预测”出来。回顾 10.2 节，罗伯特 · 梅在做鱼类种群个体数量研究的时候就曾使用这种用动力方程来进行拟合。

我们将 $x_n = kx_{n-1}(1-x_{n-1})$ 改写成 $z = xy(1-y)$ 这种形式（如图 10–5 所示），通过函数图像可以看到，这个二元函数形成的曲面是一个非常明显的非线性曲面。如果感觉这样理解还不够直观，那么就试着简化它，当 k 值一定的时候，二元函数会退化成一个一元二次函数。

以 $k = 3.8$ 为例，$y = 3.8x - 3.8x^2$（如图 10–6 所示）。虽然函数的值域确实是一个有限的值，但这显然不是一个单调函数。

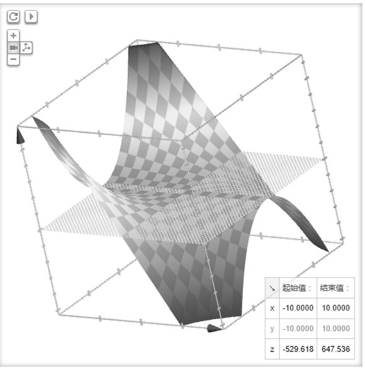


图 10–5 $z = xy(1-y)$ 图像

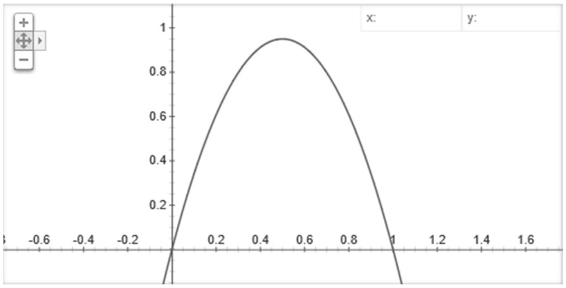


图 10–6 $y = 3.8x - 3.8x^2$ 图像

如果看不出端倪，就再展开一级。

$$y = kz(1-z)$$
$$z = kx(1-x)$$
$$k = 3.8$$

会得到

$$y = -3.8^3 \times x^4 + 2\times 3.8^3 \times x^3 - (3.8^3 + 3.8^2)\times x^2 + 3.8^2 \times x$$

x（第 1 代种群数量）与 y（第 3 代种群数量）的关系如图 10–7 所示，这是一个双峰函数图形。如果你有精力一级一级展开，就会发现这个表达式在表达 x_n 和 x_1 之间的关系时是一个“不稳定”的对应关系。如果展开到 x_3 的表达式，就是一个关于 x_1 的从 8 次项到 1 次项俱全的高次函数。x_4 则是一个关于 x_1 的最高次为 16 次项的函数……复杂程度之高，在工程研究领域，哪怕是在普通的科研领域，都是要尽可能避免的。因为其讨论太过复杂，所以这种对非线性模型经过迭代产生结果的通项公式的研究，用计算机进行快速迭代来观察结果更为方便。

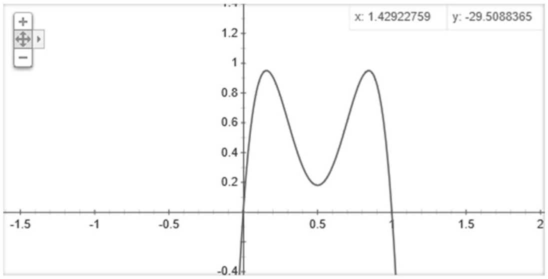


图 10–7　$y = -3.8^3 \times x^4 + 2\times 3.8^3 \times x^3 - (3.8^3 + 3.8^2)\times x^2 + 3.8^2 \times x$ 图像

在罗伯特·梅的实验中就可以看到，这种迭代性的模型都是用

$$x, f(x), f(f(x))\cdots\cdots$$

的方式进行序列的描述。结果就是，这种在初始化条件上的细微差异确实会在多次叠加之后产生难以预测的后果。这种差异的产生可以认为是“建模能力”的问题，由于使用了这种具有迭代性的非线性建模方式，导致这种预测模型最后会出现难以预测的结果。可是从另一方面来说，人类目前也没有找到更好的、可以取代这种方式的“精确”建模办法。所以，我们必须对这种现象可能带来的各种后果有心理准备。

10.4　“谋杀上帝”的拉普拉斯

我们的世界到底是决定的，还是非决定的？是可预测的，还是不可预测的？这一直是令古今中外的学者和哲学家们困惑和争论的认知领域的根本问题。300 多年前牛顿力学的诞生是科学史上一个重要的里程碑。牛顿主义的因果律和机械决定论认为：世界是可以精确预测的。根据牛顿物理学，宇宙似乎可以被想象成一台巨大的机器，其中的每个事件都是有序的、规则的和

可预测的。牛顿三大定律似乎“放之四海而皆准”——动方程有了，只要初始条件给定，物体的运动轨迹应该完全可知、可预测——直到宇宙毁灭那一天。

可以想象，这样一幅决定论（又称“拉普拉斯信条”）简单、井井有条、可预测、似乎完美无缺的理论体系和世界图景是何等诱人。它让当时的科学界欢呼雀跃，陶醉不已，以至于神学界主宰一切的“上帝”之类也想来插上一手。因此，在牛顿力学时代，宿命论、神秘主义甚嚣尘上，天才的牛顿也未能免俗。他认为，造物主实在伟大非凡，创造出的世界精妙绝伦，天衣无缝。因此，晚年的牛顿潜心研究神学。我或多或少能够体会牛顿当时的心态——这是一种顶到理论“天花板”时的心情。如果是我，我也同样会猜测“天花板”上面有什么，但我真的看不到，我只能去猜。于是，“天花板”下面的我就只能从“天花板”下面的各种迹象中努力推测“天花板”上面有什么样的事物存在。这其实已经接近当时科学的边界了。再往前走是什么？再往前走就是人类无法量化甚至无法观测的事物和现象了。那还是科学吗？

牛顿走了，拉普拉斯[83]（如图 10–8 所示）来了。拉普拉斯也醉心于牛顿力学完美的理论体系，他把万有引力定律应用到整个太阳系，研究太阳系及其他天体稳定性的问题，被誉为“天体力学之父”。不过，和牛顿不一样的是，拉普拉斯没有将功劳归于“上帝”，而是把“上帝”赶出了宇宙。

图 10–8 拉普拉斯

拿破仑看过拉普拉斯所写的《天体力学》[84]一书之后，奇怪其中为何对“上帝”只字未提。拉普拉斯自豪地说了一句话，令拿破仑目瞪口呆——“我不需要上帝这个假设！”（I Had No Need of That Hypothesis!）

在拉普拉斯眼里，如果一个智能生物能确定从最大天体到最轻原子的运动的现时状态，就能按照力学规律推算出整个宇宙的过去状态和未来状态。后人把他假定的“智能生物”称为“拉普拉斯妖”（Démon de Laplace）。老实讲，我对天主教的教义不甚理解，也不能证明“上帝”存在或者不存在，同样不知道这个“拉普拉斯妖”存在或不存在，可是从两位泰斗级的科学家这种多少令人觉得有些反常的举动中，我却感受到他们走到科学探索边界时的困惑。

现在的人对拉普拉斯的观点有两种不同的态度。一种态度是说，拉普拉斯还是没有跳出经典力学理论的误区，仍旧用经典力学理论来解释很多其不适用的现象，“拉普拉斯妖”根本就是无稽之谈。另一种态度显得略微温和一些，假设世间的万事万物都遵从这种理论，也就是说，控制人类情绪的多巴胺、内啡肽，以及其他各种荷尔蒙激素，甚至组成整个宇宙万事万物的每

83 皮埃尔–西蒙·拉普拉斯（Pierre–Simon Laplace，1749 年 3 月 ~ 1827 年 3 月），法国分析学家、概率论学家和物理学家，法国科学院院士。

84 第一卷（包括 2 册）和第二卷（包括 3 册）于 1799 年出版，内容包括理论力学原理、天体力学的基本问题、吸引理论和均匀流体自转时的平衡形状、海潮和大气潮理论，以及岁差和章动、月球天平动和土星环运动。第三卷于 1802 年出版，分为 2 册，内容包括一阶二阶摄动公式，各大行星（水星到天王星）的球坐标分析公式及有关问题，以及月球运动方程的积分方法和各种主要摄动项。

一个微小粒子，都需要用微分方程进行描述，并在某个时间给定一个初始条件，其他事情的发生都是确定的、可预测的或可向前回溯推测的。

拉普拉斯提出的这种理论甚至在理论层面都不太说得清其可行性。宇宙中所有的原子也在不停地发生着变化，有的是小原子聚合成大原子，有的则是大原子分裂成小原子。要解答这个问题，就要到更小的粒子上去建立方程了。现在宇宙中的原子已经达到一个大到不可思议的、人类无法把控的数量[85]。要知道，在 1 毫升水中有大约 3.34×10^{23} 个水分子，这还没有计算更小的粒子单位的数量。人类连"三体"问题都还没有很好地解决，又怎么可能解决这个问题？所以，对这件事情，最多是在纯理论层面讨论一下。即使这种事情真的存在，可能只有"神"才能把控吧！

10.5 庞加莱"不是省油的灯"

图 10–9 庞加莱

庞加莱[86]（如图 10–9 所示）是法国最伟大的数学家之一，也是一位理论科学家和科学哲学家，被公认为"19 世纪末和 20 世纪初的领袖数学家"（在他之前，最近一个被称为"数学全才"的数学家是高斯）。1887 年，年仅 32 岁的庞加莱被选为法国科学院院士，1906 年成为法国科学院院长，1909 年入选法兰西学术院。

庞加莱一生的成就很多。由于他在研究 10.4 节我们提到的三体问题时发现了混沌现象，更使他成为混沌论的开创者。

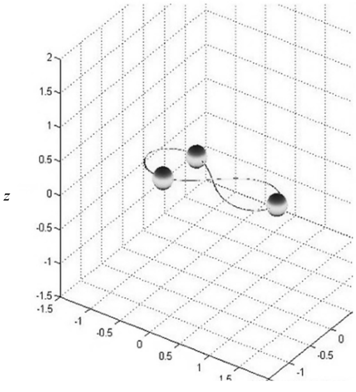


图 10–10 三体问题

三体问题是天体力学诸多问题中的一个基本力学模型，是指 3 个质量、初始位置和初始速度都是任意的可视为质点的天体，在相互之间万有引力的作用下的运动规律问题（如图 10–10 所示）。三体问题其实是"*N* 体问题"研究中 $N=3$ 时的一个特例。所谓 *N* 体问题，是指研究在三维空间中给定 *N* 个质点，假设它们之间只存在万有引力作用的情况下，当给定初始位置和速度时它们会如何运动这样一个问题。

三体问题的研究内容并不神秘，怎么看都觉得只是普通的物理力学问题。作用于其中一个质点 Q_i，那么 Q_i 收到的合力就可以表示为

85 一说为人类所见的宇宙中原子数量约为 $10^{70}\sim10^{80}$ 个。

86 儒勒・昂利・庞加莱（法语：Jules Henri Poincaré），1854 年 4 月~1912 年 7 月。

$$\sum_{j=1}^{n} F_{ij} = \sum_{j=1}^{n} \frac{Gm_i m_j}{r_{ij}^{\ 2}}(r_j - r_i),(j \neq i)$$

这就是用万有引力公式推导过来的，没什么稀奇。其中，m_i 是质点 Q_i 的质量，m_j 是天体质点 Q_j 的质量，r_{ij} 是质点 Q_i 和质点 Q_j 的距离，r_j 和 r_i 分别是质点 Q_j 和 Q_i 的矢量坐标，G 是万有引力常数，n 是天体质点数量。

3 个质点就有 3 个 F_{ij} 的表达式，每个 F_{ij} 作用在其对应的天体质点 Q_i 上，会产生一个关于下一时刻位移的表达式，这个表达式最终分解到 x、y、z 这 3 个坐标轴上。那么，对于天体质点 Q_i，会得到如下 3 个联立的表达式：

$$\left.\begin{aligned} m_i \ddot{x}_i &= \sum_{j=1}^{n} \frac{m_i m_j}{r_{ji}^3}(x_j - x_i) \\ m_i \ddot{y}_i &= \sum_{j=1}^{n} \frac{m_i m_j}{r_{ji}^3}(y_j - y_i) \\ m_i \ddot{z}_i &= \sum_{j=1}^{n} \frac{m_i m_j}{r_{ji}^3}(z_j - z_i) \end{aligned}\right\}(j \neq i, n = 3)$$

3 个质点，共有 9 个微分方程。

二体问题早在牛顿时代就已被完满解决，三体问题至今仍悬而未决，且一直是人们关注的焦点。9 个微分方程要想得到解析解，简直让人绞尽脑汁。庞加莱将此问题先简化成所谓“限制性三体问题”（如图 10–11 所示）。

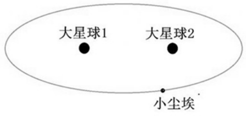


图 10–11　限制性三体问题

限制性三体问题是三体问题的特殊情况。当所讨论的 3 个天体中有 1 个天体的质量与其他 2 个天体的质量相比小到可以忽略时，这样的三体问题就称为限制性三体问题。首先，我们把小天体的质量 m_3 看成无限小，就可以不考虑它对 2 个大天体的作用。这样，2 个大天体便按照开普勒定律[87]，绕着它们的质量中心做稳定的椭圆运动（不考虑抛物线和双曲线的情形）。然后，我们再来考虑在小天体的质量 m_3 有限时，在 2 个大天体 m_1 和 m_2 的重力场中的运动。也就是说，将小天体对大天体的作用忽略不计，只考虑大天体对小天体的引力。如此简化，原来的 9 个微分方程组变成了只有 3 个变量的微分方程组。航天科学家常用限制性三体问题，研究在月球、地球引力的作用下，人造卫星、火箭及各种飞行器的运动规律。

不过，即使简化成 3 个微分方程，只有 3 个变量，也仍然无法求出精确解。庞加莱意识到，要解决问题，必须想出新的办法，不能“在一棵树上吊死”。既然无法求出精确解，那就放弃寻找精确解，开始定性地研究解的性质。庞加莱运用渐近展开与积分不变性的方法，定性研究小尘埃的轨道（如图 10–12 所示）。他深入研究小尘埃在所谓“同宿轨道”和“异宿轨道”附近的

87　德国天文学家开普勒提出的关于行星运动的三大定律。这里是指开普勒定律经过调整后解决二体问题的精确解。

行为，但一直没有得到令他满意的结果，最后不得不在 1888 年 5 月，即瑞典国王奥斯卡二世悬赏求解 *N* 体问题的比赛截稿之前提交了他的论文。

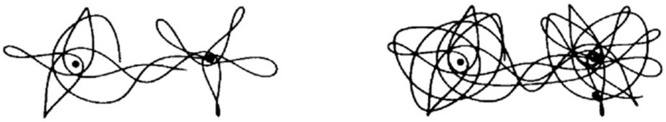


图 10-12　小尘埃的轨道

奥斯卡二世悬赏的评审团成员包括当时 3 位鼎鼎有名的数学家：法国数学家查尔斯·赫米特[88]，德国数学家卡尔·魏尔斯特拉斯[89] 和他的学生——瑞典数学家米塔-列夫勒[90]。尽管庞加莱没有完全满足奥斯卡二世悬赏的要求，没有解决 *N* 体问题，但他 160 页的论文仍然令评审团的 3 位数学巨匠兴奋无比。他们认为，庞加莱对三体问题的研究取得了重大突破，太阳系的相对稳定得到了确认。维尔斯特拉斯在给米塔-列夫勒的信中写道："请告诉您的国王，这个工作不能真正视为对所求的问题的完善解答，但是它的重要性使得它的出版将标志着天体力学的一个新时代的诞生。因此，陛下预期的公开竞赛的目的，可以认为已经达到了。"奥斯卡二世也高兴地把"奥斯卡奖"——2500 瑞典克朗和一枚金质奖章授予了庞加莱。

1889 年冬天，评审团准备将庞加莱的论文在数学杂志上发表。文章已经印好，而且送到了当时最有名的一些数学家那里。就在这时，负责校对的一位年轻数学家发现文章中有一些地方的证明不够清楚，建议庞加莱补充一段解释。于是，庞加莱开始重新深入研究这一部分。

庞加莱发现，越是深入研究小尘埃的轨道在奇点附近的性质形态，问题就越多。情况有些类似 80 多年后 MIT 的气象学家洛伦茨（10.1 节提到过）面对的困境。当然，庞加莱不如洛伦茨幸运，没有条件在计算机屏幕上显示奇异吸引子的曲线。但是，庞加莱却以他超乎常人的思维和想象能力在自己的头脑中构造出了限制性三体问题的某些奇特解的雏形。从解的奇特行为中，庞加莱看到了当今人们所说的"混沌现象"。但局限于当时的经典世界观，他未能完全理解得到的结果，只能带着迷惑感叹："无法画出来的图形的复杂性令我震惊！"

庞加莱意识到，既然解的图形复杂得无法画出来，那么在原来的论文中，不仅包含像那个校对论文的年轻数学家所说的那种"证明不够清楚"的小问题，甚至可能包含"错误"。于是，他赶快通知米塔-列夫勒，收回已经印好的杂志并予以销毁，同时大刀阔斧地修改和赶写论文。直到第二年，1890 年 10 月，庞加莱长达 270 页的新版论文才发表出来。

88　查尔斯·赫米特（Charles Hermite，1822 年 12 月 ~ 1901 年 1 月），法国著名数学家，在数论、不变量理论、正交多项式等数学基础研究方面有重大贡献，以他的名字命名的重要概念有"Hermite 矩阵"等。

89　卡尔·特奥多尔·威廉·魏尔斯特拉斯（1815 年 10 月 ~ 1897 年 2 月），德国数学家。

90　米塔-列夫勒（Mittag-Leffler，Magnus Gustaf，1846 年 3 月 ~ 1927 年 7 月），瑞典数学家。

庞加莱庆幸对论文做了这一重要的修正。而且，正是这个“错误”，使庞加莱重新研究和思考方程的解的状况，改正了一个稳定性定理，并最终使他发现了同宿交错网。

庞加莱发现，即使是对简化了的限制性三体问题，在同宿轨道或者异宿轨道附近，解的形态也会非常复杂，以至于对给定的初始条件，几乎没有办法预测当时间趋于无穷时这个轨道的最终命运。而这种对轨道的长时间行为的不确定性，就是我们现在所说的混沌现象。

混沌现象的产生原因是：在解析解无法给出的情况下在迭代中使用数值解，并由于数值解精确程度的问题产生的求解收敛到不一致的值的问题。只要使用迭代的方式建模，并使用精度有限的数值作为初始值，这种现象就无法避免。直到今天，这个问题也是科学家们热衷研究的一类问题。

10.6　未知居然还能做预测

说了这么多，大概会勾起许多人的悲观情绪。混沌既然是一种客观现象，而且人类到现在都无法克服它，是不是意味着不能做预测了呢？

实际上，有这种情绪的人放大了混沌对预测的影响范围。要知道，在认知归纳的过程中，只要人们把足够多的因素放到模型中，经过相对精确的测算，进行相对短时间内的预测还是比较现实的。

不是因为别的，只是因为在短时间内，当初始值的给定误差足够小的时候，它的结果还不至于“偏”出太远。所以我们看到，现在很多基于科学的归纳认知方式进行的预报在短期内基本能够应验，例如未来 1 小时天气预报、未来 24 小时天气预报，以及近距离的交通流量预报（避免拥堵）等。在目前的计算能力和观测条件下，这些预测的准确性比较高。

即使不能以非常精确和确实的数字给出预测结果，也可以用基于统计的概率性结论给出预测结果，这同样对人类的生产和生活有意义，因此完全没有必要悲观。

10.7　本章小结

这一章的小结是非常难写的。混沌理论从正式提出到现在已经过去半个多世纪了，人们一直在尝试研究它。

在研究混沌现象的过程中，人们都是通过微分方程组或者动力方程去解析一个系统，而在这个过程中自然无法克服被引入的非线性因素及它们彼此叠加产生的效应。这也就导致用这种解析方式设立的系统模型会对初值非常敏感，当初值的某个向量有很小的变化时，在时间足够久的情况下，必然会引发预测的极大不准确性。

这不禁让我想起“随机事件”这种同样让人难以把握的事情。

随机事件是一种在观测中可能发生，也可能不发生的事情。随机事件有 3 个特点：第一，可以在相同的条件下重复进行；第二，每个试验的可能结果不止 1 个，并且能事先预测试验的所有可能结果；第三，进行一次试验之前不能确定哪一个结果会出现。

在我看来，不论是“随机事件”还是“混沌理论”，都是一种对事物发生规律的解释，而且都是在认知达到一定程度，在有限的感知维度中对世界作出的解释理论。我们再详细讨论一下对这些事情的认知方式。

就拿随机事件来说，人们看到一个六面的骰子扔出去，得到 1 ~ 6 之间的点数。随着投掷次数的增加，每个点数出现的次数都会占总投掷次数的 $\frac{1}{6}$ 左右。确实，对某次投掷来说，得到的点数是不确定的，但是在大量投掷实验中，统计出来的比例是相对确定的。这种对于随机事件的研究就有了它的科学性——精确性、体系性、应验性都很好。

也有一派观点指出，这种“随机性”其实不是真的随机，因为在骰子被丢出去的一瞬间，如果能将当时的空气温湿度、空气密度、空气流动速度、重力加速度，骰子自身所有的描述参数（例如骰子的质量、骰子的体积、投出的角度、投出点距离地面的高度、投出时的速度、骰子投出时旋转的方向和速度），地面与骰子接触时因为碰撞所产生的弹射及能量损失变化的描述，以及骰子与地面发生相对摩擦时所损失能量的描述等因素，都列入一个完整的微分方程组进行求解——如果想让基于这个观测所归纳出来的结论尽可能准确，就要尽可能多地引入与此相关的因素——只要这些值的测量都足够精确，模型建立适当，那么对某一次投出的结果预测就还是高度精确的。可是，在我们不知道这些条件的全部或者部分的情况下，如果还想对这类事件进行描述，就可以使用完全基于统计的认知方式。这不得不说是人类在认知这件事情上，在高度精确和高度简洁两个极端作了一个折中式的妥协，或者说，无边无际的未知和有限的人类大脑给出了这样一个适合“懒人”的办法。因为人类的认知能力永远无法把“一切”可以引入的因素全都引入，所以，即使我们认识到“应该引入”，但由于主客观原因无法引入的因素也都会在我们难以量化其影响的情况下影响着认知结果——真是个可怕的结论。

混沌现象呢？也不例外。

不论是在气象、海洋、天文等领域，还是在人类社会、动物种群这类有生命的组织形式中，只要观测到的对象是大量且彼此有影响的，最后得到的描述解大都对初始条件敏感。初始条件的获得也基于人类的测量。既然是测量，就必然会有误差，而误差的客观存在也就导致一种结果——只要时间足够久，出现的长期“结果”就一定是未知的。

在我们以这类方式建模并进行大规模科学计算的时候，也免不了会遇到这样的问题。人类在无限逼近认知真实方面还有很长的路要走，不论是乐观还是悲观，尽力迎接挑战就好。

第 11 章　算法学

算法学几乎是计算机领域的必学课程。像前端工程师（以 HTML、CSS、JavaScript 为工作对象语言）和 DBA（数据库管理员，以 SQL 和 shell 命令作为工作对象语言）这样不用编写完整程序代码的人员，以及不用编写底层代码的 IT 工作者，也要了解算法的基本原理，以获得解决日常工作中出现的一些简单问题的思路。

算法学应该说是从计算机组成原理中慢慢进化出来的一种研究如何“控制”计算机的学科。在用电子管和二极管制造出第一台电子计算机以后，虽然经过几代的技术演进，到现在也只不过是集成度更高了而已，在底层的数理计算逻辑上没有本质变化，仍然在用“与”（AND）、“或”（OR）、“非”（NOT）来对加法进行模拟（如表 11-1 所示）。

表 11-1　二变量与或结果表

A	*B*	*A* 与 *B*	*A* 或 *B*	*A* 非	*B* 非
1	1	1	1	0	0
1	0	0	1	0	1
0	1	0	1	1	0
0	0	0	0	1	1

通过逻辑门单元的“与”、“或”、“非”可以实现加法和减法（实质上还是加法），其他任何计算在底层都是用加法和减法来实现的——是的，任何计算。

让计算机按照人类想定的方式去运行并得到结果，就是我们刚才所说的“控制”。计算机非常愚笨，完全没有主动思考的能力，每个操作都要由人来告诉它控制的逻辑，而它自己也只有遵守的份儿。那么，如何控制更为合理有效就是一个分支学科所要研究的范畴了。算法研究的事情就是：让计算机按照何种程序逻辑才能尽可能高效且正确地运算出结果。

11.1　离散的世界

电子计算机使用的寄存器、内存单元、磁介质存储单元、半导体介质存储单元都有一个共同的特性，那就是每个基本单元只有“导通”（1）和“截止”（0）两个状态，这是由最初的电

子计算机的物理实现基础所导致的。所以，一切计算机中的数理逻辑也只能用 1 和 0 来表示。如果用 1 位（bit）来表示数据含义，最多只能表示 2 个不同的数据含义；2 位就是 4 个；32 位就是 4294967296 个，这也是 32 位电子计算机在不开启 PAE[91] 模式的情况下所支持的最大寻址空间——4G；64 位就是 18446744073709551616——16E[92]（1844.7 亿亿），在现阶段的民用领域是非常巨大的数字了。

虽然 18446744073709551616 这个数字非常巨大，但在表示实数这样稠密的数据集合时仍然捉襟见肘。根据实数稠密性原理，任意两个不相等的实数之间都有无穷多个实数，也就是说，不论未来电子计算机的寻址能力是 128 位、256 位还是 32768 位，对于描述整个实数域来说仍远远不够。所以，小数（浮点数）的表示只能用类似科学计数法的方式用“有效数字”乘以“10 的 n 次方”表示。

在电子计算器上，通常用科学计数法表示巨大的计算结果（如图 11-1 所示）。根据微软的相关技术白皮书，双精度 Double 变量以带符号的 IEEE 64 位（8 字节）双精度浮点数形式存储，负值的取值范围为 –1.79769313486231570E+308 到 –4.94065645841246544E–324，正值的取值范围为 4.94065645841246544E–324 到 1.79769313486231570E+308[93]。虽然对无限稠密的实数来说，这保留下来的区区 18 位有效数字和理想的实数域有着天壤之别，可对于人类现阶段的绝大部分工商业活动来说已经足够了。与有损压缩中的哲学思维一样，这些即使舍弃也对人的感知影响极为微小的因素是可以在考虑“性价比”的前提下做适当取舍的。这种思维方式在数据科学领域的实践过程中始终如一。

图 11-1　电子计算器上的科学计数法

在以有限去仿真无限，以离散去仿真连续的情况下，永远存在这种问题。所以，对那些我们无法精确感知的，感知了也无法把控的，把控了也对提高效益没有意义的场合，确实需要斟酌精度提升的必要性。

在离散数学里有一个非常重要的概念叫作“笛卡儿乘积”（Cartesian Product）。笛卡儿乘积实际上是离散的面积、体积的概念。假设有一个集合 X，有 n 个元素 $\{x_1, x_2,\cdots, x_n\}$；另外有一个集合 Y，有 m 个元素 $\{y_1, y_2,\cdots, y_m\}$：

91　物理地址扩展（Physical Address Extention），是基于 x86 服务器的一种功能，能使运行该模式的计算机支持 4GB 以上的物理内存。

92　1Y=1024Z，1Z=1024E，1E=1024P，1P=1024T，1T=1024G，1G=1024M，1M=1024K，1K=1024。

93　参见 Visual Studio .NET 2003 数据类型说明部分。

$$X \times Y =$$

$$\{$$

$$x_1y_1, x_1y_2, \cdots, x_1y_m,$$

$$x_2y_1, x_2y_2, \cdots, x_2y_m,$$

$$\cdots$$

$$x_ny_1, x_ny_2, \cdots, x_ny_m,$$

$$\}$$

用一个矩阵表示这个笛卡儿乘积的结果 $x_2y_1, x_2y_2, \ldots, x_2y_m$

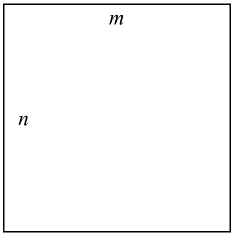
x_1y_1	x_1y_2	$\cdots$	x_1y_m
x_2y_1	x_2y_2	$\cdots$	x_2y_m
$\cdots$	$\cdots$	$\cdots$	$\cdots$
x_ny_1	x_ny_2	$\cdots$	x_ny_m

通过观察可以发现，在这样一个矩阵中，每个元素的和相当于整个矩阵的面积，也就是这样一个表达式

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_i y_j$$

与长和宽分别为 n 和 m 的矩形的面积公式

$$S = mn$$



相比。这种笛卡儿乘积同样是把正交的 2 个维度做了一种二重循环式的叠加，而且这种叠加在笛卡儿乘积中可以应用到多个正交维度——扩展到 3 个、4 个或更多个维度。这种离散的视角在使用计算机解决问题的过程中非常普遍。在计算机中，建立包含很多描述对象的模型时也会用到这种视角。例如，在关系型数据库系统中，多个表之间使用 JOIN 连接，实际上相当于把一张一张的表看成一个一个独立的维度，进行 JOIN 的过程相当于在这种相乘之后的面积上进行谓词过滤或聚合操作（Group By）。

11.2　成本的度量

理想的算法要求计算结果“准确无误”。在模型的解空间中，如果元素是离散的，就通常是计算机擅长的范围。例如，经典的“八皇后问题”[94]就是要求出所有符合要求的解（如图 11–2 所示），不重、不漏、不错。在这种要求下，算法的优劣就只和处理的效率有关，也就是更快的算法更好。

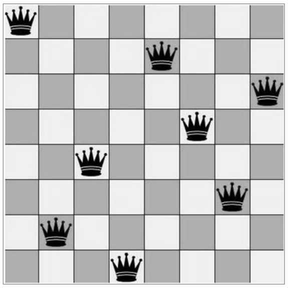


图 11–2　八皇后问题的一组解

在算法领域对算法效率度量的描述方式记作 O。用这个描述符号表示算法在处理对象的数量 n 变化的时候所产生的处理时间成本，称为时间复杂度。好的算法在处理数据时所花费的时间随着处理对象 n 的增加而线性增加，甚至会有比线性增加效率更高的方式；不好的算法则会在 n 增加时所花费时间呈指数级增加抑或更高。

常见的算法的时间复杂度有 $O(\log n)$、$O(n)$、$O(2^n)$、$O(n^2)$、$O(n\log n)$ 等（如图 11–3 所示）。

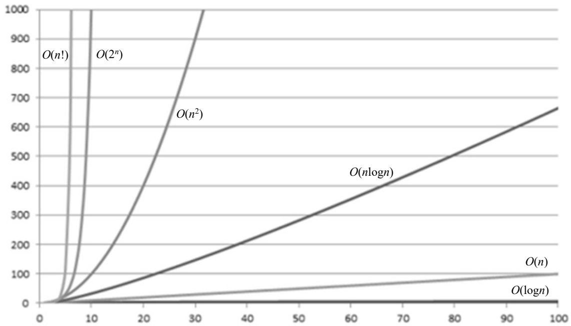


图 11–3　各种时间复杂度

例如，写一个从具有 n 个节点的单向链表（如图 11–4 所示）中找出某个确定值的算法，就需要遍历链表了。当这个查找对象在任何一个点上的概率均等的时候，平均查找次数是多少？这很容易计算：

$$\frac{1+2+\ldots+n}{n}$$

也就是

$$\frac{n+1}{2}$$

这种情况是和 n 成正比的，所以记作 $O(n)$。

94　在一个国际象棋棋盘上放置 8 个皇后，让它们彼此之间不能直接互相攻击。皇后在国际象棋规则中可以横、竖、斜走无限格。

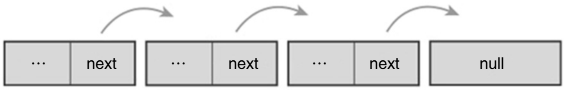


图 11-4 单向链表

再如，在用平衡二叉树对某一确定值的数据进行查找的算法中，由于每次查找都会在访问一个树节点的时候“拐弯”，放弃对本层以下其他树节点及其子节点的查找，从而“最快”接近目标节点，而这种深入的次数只与层数有关，所以它的时间复杂度为

$$O(\log n)$$

这也是在熵最大的情况下查找的平均次数。需要强调的是，这是一种将场景简化为纯学术层面的讨论方式。在实际的算法编写过程中，数据在寄存器、内存、磁盘上的处理时间成本是非常悬殊的，所以这种讨论通常存在于学术层面，都是在理想条件下进行的讨论，这是它的局限性。不过，即使这样，也不能说它是错误的，因为它是一种被学术界广泛认可的、理论指导意义很高的度量方式，而且描述简洁方便。

在具体的生产中，还是建议分两步走。第一步，在刚刚所列举的纯模型层面去做一个规模性的估算，尽量避免比线性增加 $O(n)$ 时间复杂度更高的情况出现。第二步，以生产场景中的处理规模与硬件设备做实测或等比缩放的实测。

11.3 穷举法——暴力破解

这种方式最原始、最笨重，但也最简单，千万不要觉得自己如果想到用这种方法去解决问题就万分羞赧，因为它同样是一种人与生俱来的解决问题的思维方式。

在少量数据的场合，以及硬件资源极为丰富的情况下，穷举法反而是最经济的办法，因为它消耗的资源是最廉价的。否则，要解决问题还需要进行一系列的建模、优化、测试、调整、论证、反复比对等过程，这种时间成本及在人工上的消耗很难忽略不计。

这种场景对于模型的解空间有限离散的情况最为合适。

经典的“八皇后问题”的求解（代码如下）就是穷举法，反正计算规模是非常小的——8 的 8 次方，只要 16777216 次计算就可以完成，这对现代高速的计算机来说简直是“秒得”的结果。这种情况下，我在写程序时也会优先考虑直接穷举，因为我不确定在保证自己不出错的情况下进行优化，得出解的速度是不是更快。

```
def conflict(state, nextX):
    nextY = len(state)
    for i in range(nextY):
        if abs(state[i]-nextX) in (0, nextY-i):
            return True
```

```
    return False

def queens(num, state=()):
    for pos in range(num):
        if not conflict(state, pos):
            if len(state) == num-1:
                yield (pos, )
            else:
                for result in queens(num, state+(pos,)):
                    yield (pos, ) + result

def prettyprint(solution):
    def line(pos, length=len(solution)):
        return '. ' * (pos) + 'X ' + '. '*(length-pos-1)
    for pos in solution:
        print line(pos)

if __name__ == "__main__":
    i = 0
    for solution in queens(8):
        i+=1
        print 'solution:' + str(i) + '\n'
        prettyprint(solution)
```

这种经典算法与数据结构的例子网上有很多，其中 `queens(8)` 这一部分是可以修改的，“8”代表的是 8×8 的棋盘，放置 8 个皇后。如果改成 `queens(16)`，就是在一个“改造”过的 16×16 的棋盘上求解一个十六皇后问题了。

为了展示这种穷举法在八皇后问题扩展上的“缺陷”，需要对刚刚的程序做一下简单的改造，把它改造成一个能够处理“N皇后问题”的程序。

程序的前部改为：

```
import sys
number = int(sys.argv[1])
```

程序的后部改为：

```
if __name__ == "__main__":
    i = 0
    for solution in queens(number):
        i+=1
    print 'total test:' + str(pow(number,number)) + '; total solution:' + str(i) + '\n'
```

这样就可以打印出所有的尝试次数和有效解的个数，并且，只在终端输出个数，而不输出具体解出的盘面。

接下来，就可以用命令行从 1 开始逐步尝试。

```
[root@ann science]# time python eight.py 1
total test:1; total solution:1

real 0m0.011s
user 0m0.003s
sys  0m0.008s
[root@ann science]# time python eight.py 2
total test:4; total solution:0

real 0m0.010s
user 0m0.006s
sys  0m0.004s
[root@ann science]# time python eight.py 3
total test:27; total solution:0

real 0m0.011s
user 0m0.008s
sys  0m0.005s
[root@ann science]# time python eight.py 4
total test:256; total solution:2

real 0m0.009s
user 0m0.003s
sys  0m0.006s
[root@ann science]# time python eight.py 5
total test:3125; total solution:10

real 0m0.031s
user 0m0.017s
sys  0m0.014s
[root@ann science]# time python eight.py 6
total test:46656; total solution:4

real 0m0.011s
user 0m0.007s
sys  0m0.004s
[root@ann science]# time python eight.py 7
total test:823543; total solution:40

real 0m0.053s
user 0m0.018s
```

```
sys  0m0.035s
[root@ann science]# time python eight.py 8
total test:16777216; total solution:92

real 0m0.030s
user 0m0.023s
sys  0m0.007s
[root@ann science]# time python eight.py 9
total test:387420489; total solution:352

real 0m0.081s
user 0m0.077s
sys  0m0.004s
[root@ann science]# time python eight.py 10
total test:10000000000; total solution:724

real 0m0.320s
user 0m0.316s
sys  0m0.004s
[root@ann science]# time python eight.py 11
total test:285311670611; total solution:2680

real 0m1.723s
user 0m1.717s
sys  0m0.006s
[root@ann science]# time python eight.py 12
total test:8916100448256; total solution:14200

real 0m10.202s
user 0m10.185s
sys  0m0.016s
[root@ann science]# time python eight.py 13
total test:302875106592253; total solution:73712

real 1m3.286s
user 1m3.273s
sys  0m0.009s
[root@ann science]# time python eight.py 14
total test:11112006825558016; total solution:365596

real 7m18.573s
user 7m18.562s
sys  0m0.017s
```

再往下测试的话时间就非常长了。在这里我只测试到 14 个皇后的情况（如表 11–2 和图 11–5 所示）。这种单线程的计算效率几乎完全依赖单 CPU 核心的主频。

表 11–2 *N* 皇后求解时间表

盘面宽度	穷举次数	解 个 数	时间（秒）
1	1	1	0.003
2	4	0	0.006
3	27	0	0.008
4	256	0	0.003
5	3125	10	0.017
6	46656	4	0.007
7	823543	40	0.018
8	16777216	92	0.023
9	387420489	352	0.077
10	10000000000	724	0.316
11	285311670611	2680	1.717
12	8916100448256	14200	10.185
13	302875106592253	73712	63.273
14	11112006825558016	365596	438.562

这种试探次数的增加，在 *N* 皇后问题场景中随着盘面宽度 n 的增加而迅速增加（如图 11–5 所示）。显然，这个时间复杂度是 $O(n^n)$，几乎是在二维空间解中效率最差的一种情况了。

测试的时间目测也是按照 $O(n^n)$ 来增加的（如图 11–6 所示）。为了验证我们的观点，下面用“叠加法”来估算单次试探所需要的时间。由于单次试探的时间太短，所以任何一次测量都会产生非常大的误差。我们只看盘面宽度为 8 以前的各次测试所花费的时间就可以了。

这两组数据几乎看不出任何联系，波动太大，所以不建议参与统计（如图 11–7 所示）。

盘面宽度和单次试探时间的关系像是一条渐近线，随着盘面宽度的增加，单次试探时间无限趋近于 0（如表 11–3 和图 11–8 所示）。实际上，这个试探时间无论如何是不可能为 0 的。

表 11–3 *N* 皇后求解盘面宽度为 *N* 与 *N*–1 时的单次试探时间比

盘 面 宽	单次试探时间（秒）	盘面宽度为 *N* 与 *N*–1 时的单次试探时间比
9	1.9875E–10	0.144977364
10	3.16E–11	0.158993344
11	6.01798E–12	0.190442413
12	1.14232E–12	0.189817095
13	2.08908E–13	0.182881075
14	3.94674E–14	0.188922486

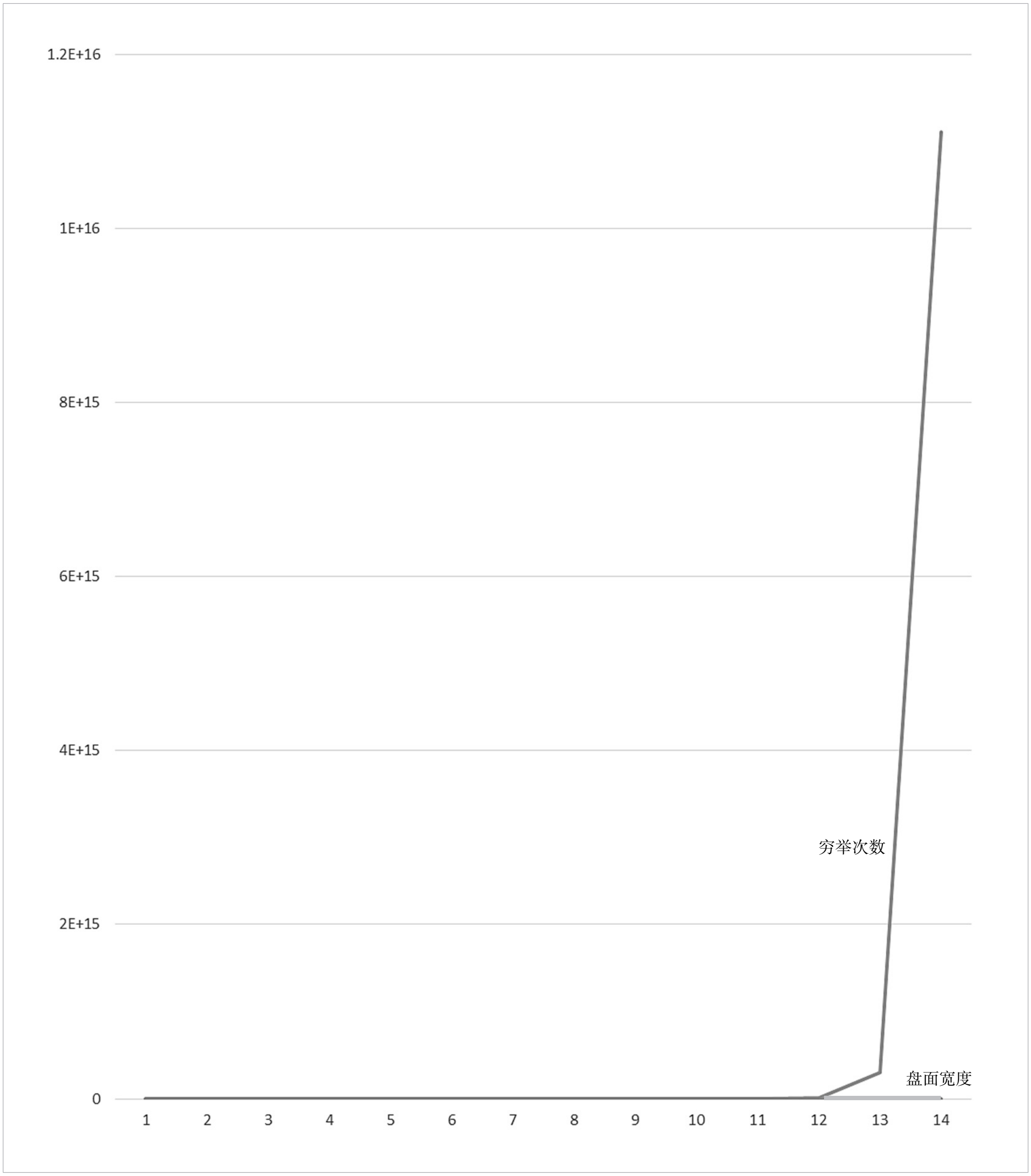


图 11–5　盘面宽度–穷举次数

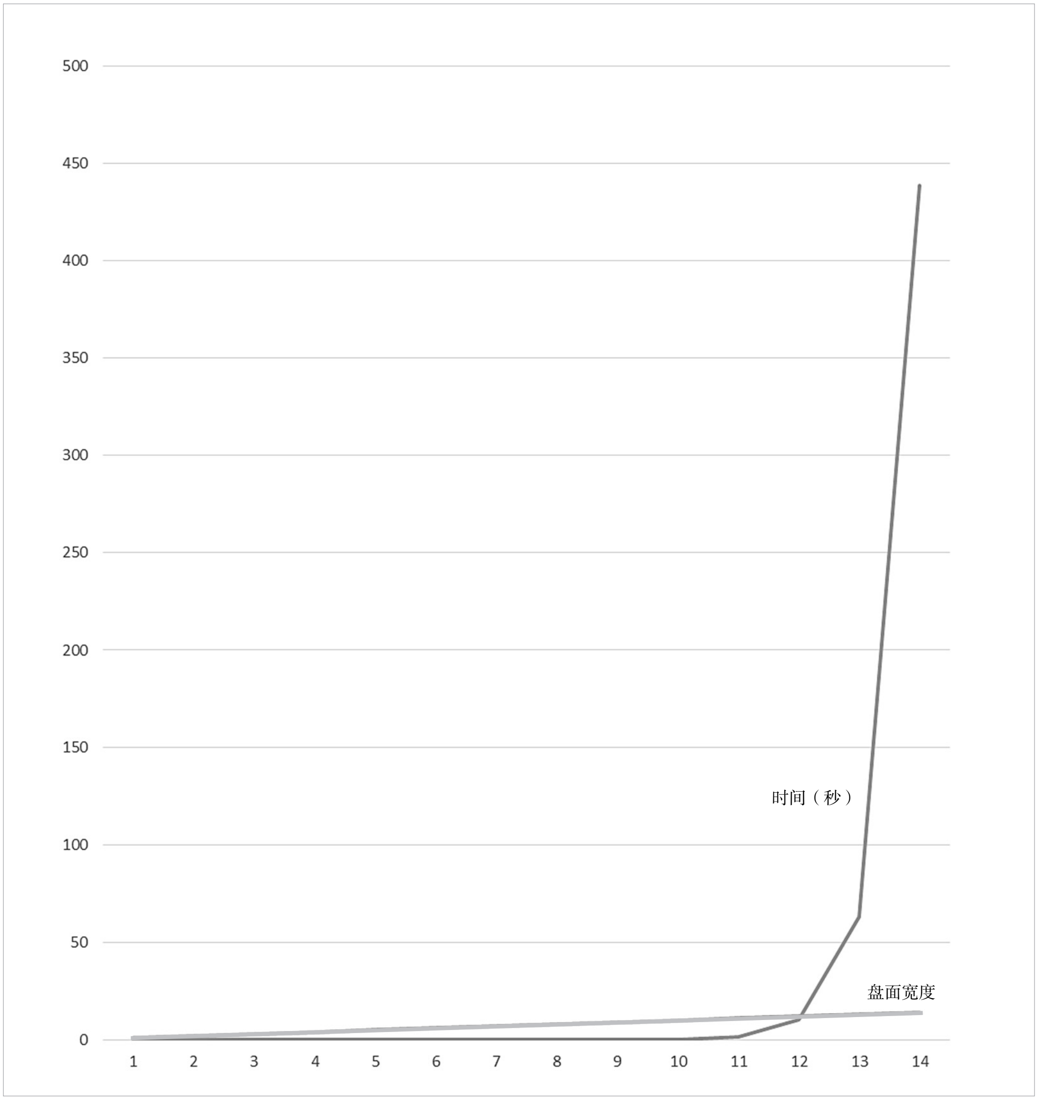


图 11-6　盘面宽度-时间（秒）

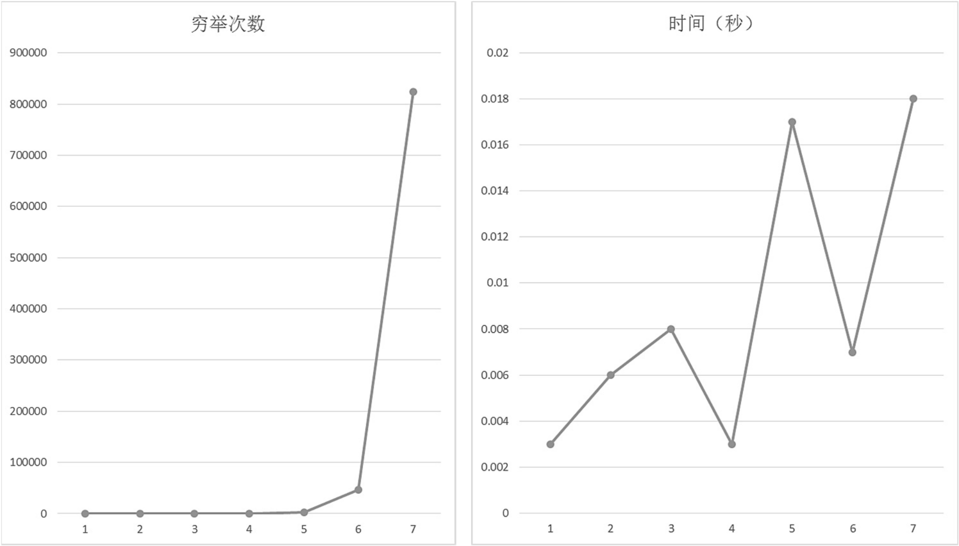


图 11–7　穷举次数与花费的时间（秒）

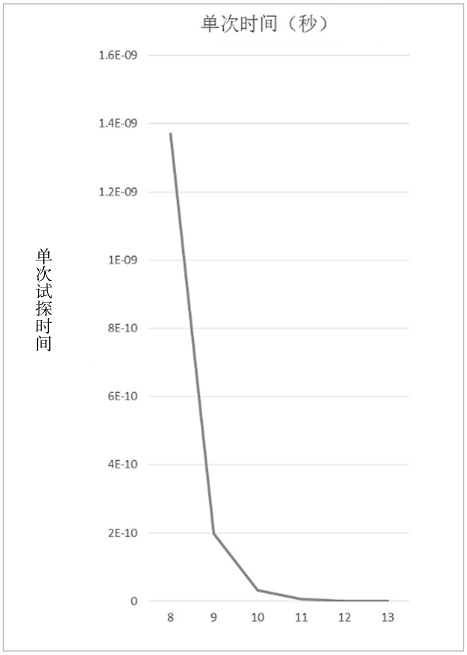


图 11–8　盘面宽度–单次试探时间

这就表明，N 每增加 1，平均每次试探的时间就会变为之前的 0.18 倍左右（如图 11-9 所示）。这样我们可以猜测：当 $N=15$ 时，单次的平均时间可能还会继续以这个比例下降。

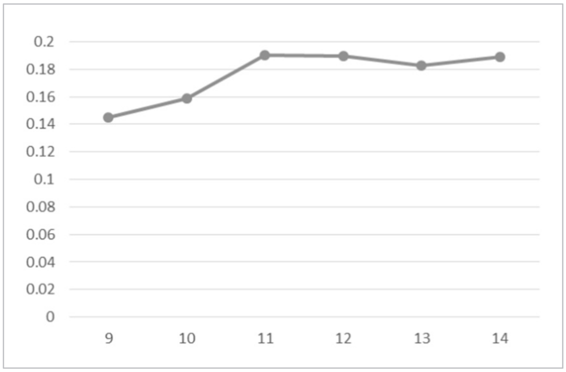


图 11-9 盘面宽度为 N 与 $N-1$ 时的单次试探时间比

$N=15$ 的情况如下。

```
[root@ann science]# time python eight.py 15
total test:437893890380859375; total solution:2279184

real 52m46.628s
user 52m46.265s
sys  0m0.094s
```

计算结果如表 11-4 所示。可以看出，平均单次试探时间下降为前一次的 0.18 倍左右。

表 11-4 N 皇后问题求解时间（$N=15$）

盘面宽度	穷举次数	解 个 数	试探时间（秒）	单次试探时间（秒）	盘面宽度为 N 与 $N-1$ 时的单次试探时间比
15	4.37894E+17	2279184	3166.625	7.23149E-15	0.183226907

在这个例子中我们发现了一个现象：虽然时间复杂度是 $O(n^n)$，但是花费的时间并非如此。在测量中也发现：单次试探的时间每次下降为之前的 0.18 倍左右。也就是说，当 $N=15$ 时，我们还没有找出这条渐近线的极限。没关系，下面这两个结论已经很重要了。

结论 1：时间复杂度的增加和实际运算的时间增加不成比例

千万不要“望文生义”，想当然地认为时间的增加和迭代次数成正比。具体的消耗时间需要通过实际的测量得到，这种方法更为准确。在其他算法场合下，可以用更大的 N 来尝试，例如 1000、5000、10000、50000……这样的序列。对这些点的测量会给那些位于这些点之间的未测量的 N 的场景提供一个参考刻度。

结论 2：如果可能，尽量测出这个极限的位置

也就是说，在合理的时间成本范围内尽量测出最后不再下降的单次试探时间。例如，连续 2 次增加 N，都发现单次试探时间不再有明显下降，即可考虑给出这样的结论：可以作为更大的 N 出现时所花费时间的评估基础。

如果要在一个庞大的分布式集群里对未知节点上的某一条数据进行查找，在没有索引的情况下，使用的同样是典型的穷举法。别以为所有的查找场景情况下都会有索引，索引的建立和维护是有成本的，无论是 OLAP 系统，还是 OLTP 系统，对索引的使用都是比较谨慎的。

再如，在没有任何提示的情况下破解一个密码也是典型的穷举法使用场景——必须一个一个试过，才会知道这个密码是否与正确的密码吻合。

穷举法适用于数据量小、硬件极为丰富或信息预处理不充分的场景。要事先根据经验或实际的时间消耗测算来预估破解的总时长，以判断是否可以使用穷举法。

11.4 分治法——化繁为简

分治法的应用在生产中更为普及，而且这种思路可以说同样是浑然天成——也是人类自己处理复杂问题的逻辑。人类在碰到一个庞大而复杂的问题时同样会想：这个问题可不可以分解成多个简单一些的小问题？每个问题获解了，整体就获解了（如图 11-10 所示）。

这种思维方式的一个好处是可以降低求解难度。在日常生活中我们知道，处理 1 个对象事物和处理 10 个对象事物的成本不是 10 倍的关系。举个现实生活中的例子，一个人自己做饭吃，翻翻电冰箱，有些鸡蛋和西红柿，炒个鸡蛋西红柿凑合凑合就行了；而如果 10 个人一起聚餐，即使有足够的鸡蛋和西红柿，也不能炒 10 倍量的鸡蛋西红柿大家一起凑合，要采买的蔬菜和肉类多了，准备工作复杂了，花费的时间成本也就高了。这里的成本关系不是线性的，这种方式对一个人来说处理的成本太高了。怎么办？最好的办法就是事先商量好，每个参加聚餐的人都带一个做好的菜——人数越多，效果越明显。这种方式的处理结果就是每个人的时间成本略有增加，但是不会把其中一个人累死——开个玩笑。

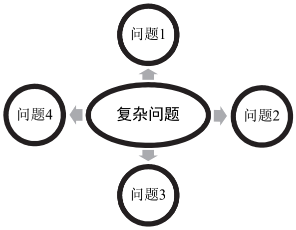


图 11-10 问题分解

在经典的单机应用场景中，也有一些算法应用了分治法。例如，二分查找、多路归并排序、大整数乘法等，都是典型的把复杂问题简化为局部与局部之间的简单问题获解进而得到全局解的思路。

这种思路在其他更宏观的场景中一样可以见到。例如，在一个 1PB 的数据集中，要查找其中的最大值。这种数据在一台计算机上怎么存储都是个问题，即使能存储，也要一部分一部分

地把数据提取到内存中进行排序，找出最大值，然后将每次迭代的最大值保留，与下一次迭代的最大值相比较，保留两者中较大的值，直到遍历 1PB 的数据为止。

这种办法几乎只是理论上可行，因为它所消耗的时间成本极其高昂。而在大型的 Hadoop 集群中，可以在几百甚至几千个节点上分别查找，每个节点只要找到本节点上的最大值，最后找出这些值中的最大值，就能获解。这种任务的加速几乎是随着集群规模的增加而线性加快的（如图 11–11 所示）。对海量数据处理来说，Hadoop 框架的解决方案本质上就是分治法的应用。

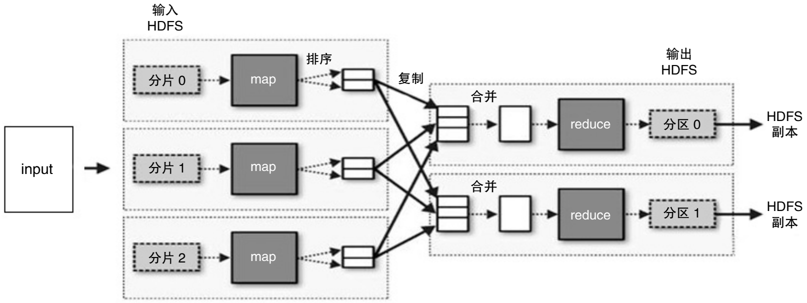


图 11–11　Hadoop 中各 map/reduce 分治执行任务

不仅如此，程序员熟悉的 OOP 技术（Object Oriented Programming，面向对象编程）和过去使用的 POP 技术（Procedure Oriented Programming，面向过程编程）的设计思路也有分治思想的深刻烙印。

在一个类的设计中，除了在必要的对外服务功能部分用 public 的方式暴露出来以外，其他的内容都在内部用 protected 方式或 private 方式进行保护（如表 11–5 所示）。

表 11–5　面向对象的不同使用权限

描　　述	当 前 类	同 PACKAGE	子 孙 类	其他 PACKAGE
public	YES	YES	YES	YES
protected	YES	YES	YES	NO
friendly	YES	YES	NO	NO

而且，面向对象的设计几乎将一切调用细节全部推给了编译器和虚拟机软件，把返回功能结果的要求推给了类和方法的提供者，OOP 语言仅关心当前编程人处理流程中最为核心的部分，而当前编程人也以同样的方式服务于其他人，颇有点“人人为我，我为人人”的社会大分工互利共赢的意思。

分治法的思路在处理很多大而复杂的问题时都可以使用，而且形式灵活，只要使用得当，确实能够“化不可能为可能”。不过要注意，这种问题的模型必须是一种可分治的模型，如果问题设计本身无法通过多个局部获解而使得全局获解，分治法的思路就行不通了。

在分布式环境中使用全局去重的命令时就无法应用分治思想。例如，在包含 N 个节点的 Hive 或者 Spark–SQL 环境中使用 DISTINCT 进行全局去重，要实现这个功能有两种方法。

第一种方法是把 DISTINCT 命令发送到所有节点。这个求解的指令发送到任何一个节点之后，做的都是同样的事情，每个节点要先拿谓词“where”在本地进行过滤。在过滤后的集合中，把每一条数据 X_i 拿到其他 $N-1$ 个节点进行询问，如果 X_i 出现在其他节点，就需要通过一种协商机制要求对方节点在输出结果集的时候对 X_i 做不输出处理。某节点为其他节点询问到的 X_i，如果已经询问其他节点并决定做输出处理时，应通知对方做不输出处理。这种方式可以保证在全局范围内以声明的时间戳为准决定最后在全局输出哪些 X_i 记录。这种相互之间的询问和应答会引发节点间大量的信息传递，并诱发网络风暴（如图 11–12 所示），导致全网瘫痪。

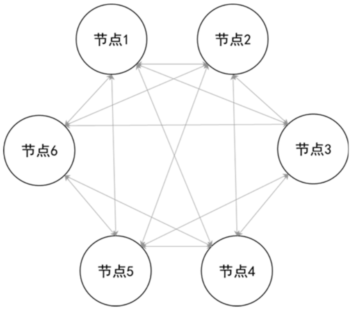


图 11–12　网络风暴

如果不这么做，就只能使用第二种方法了。把所有数据（X_i）都传送到一个节点上，在这个节点上做哈希排重（单机无序输入排重效率最高的方式）。可以看到，当局部子问题严重依赖其他局部子问题的时候，分治法就不适用了。分治法只适用于那些可以简化为使用高内聚、低耦合方式解决的问题。

11.5　回溯法——能省则省

回溯法（探索与回溯法）是一种选优搜索法，又称“试探法”，有些参考资料上也称其为“剪枝法”。回溯法的核心思路是，按选优条件向前搜索，以达到目标。当搜索到某一步时，发现原先的选择不优或达不到目标，就退回一步重新选择。这种走不通就退回再走的方法称为“回溯法”，而满足回溯条件的状态点称为“回溯点”。

这种思路的“上一步”实际上是穷举解空间的思路。在对空间进行穷举的过程中，发现按照当前的穷举路径前进，剩下的解都不可能是所求解，在这种情况下，就放弃当前穷举路径下的所有待选解并回退。所以，在绝大多数情况下都只尝试了穷举空间中的一小部分解，而不是全部解。

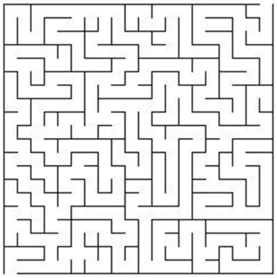


图 11–13　迷宫问题

例如，写一个算法来模拟走出迷宫的过程。这个算法可以用递归来写，也可以用非递归的遍历方法来写。如图 11–13 所示是一个 20×20 的迷宫，在迷宫的每个方格中都有 4 个方向可以“走”，除了过来的方向（走过的格子），还有 3 个方向可以“走”。那么，在遍历的过程中，理论上的计算成本为

$$3^0+3^1+3^2+...+3^{399}=3\cdot\frac{1-3^{399}}{1-3}$$

大约是 3.5×10^{190}。在这样的情况下，使用穷举法显然会消耗一段我们无法忍受的时长，而且使用分治法的思路也不明确，无法简化成“局部获解而全局获解”的问题。

但是，我们可以很容易地发现，迷宫中很多方格的 4 个方向其实并非全部有效：有的是“墙”，过不去；有的是走到某一步会发现，从这一步开始的所有解树叶节点都不用尝试了，这种情况下，会把“树”上的所有子枝全部剪掉，以节省大量的空间和时间成本，所以叫“剪枝法”也是非常形象的（如图 11–14 所示）。

剪枝法不仅可以应用在类似寻找迷宫出口的问题上，在地图导航等领域的应用及变种应用也很多，例如象棋和围棋的对弈软件就是使用类似的方式实现的。不过，近年来随着深度学习领域的理论发展和计算机计算能力的攀升，围棋对弈软件更多使用深度神经网络辅以强化学习的手段来开发。关于深度神经网络解决问题的核心思路，会在 11.9 节专门讨论。

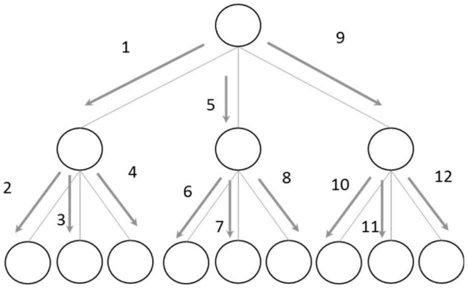


图 11–14　三叉树深度优先遍历

11.6　贪心法——局部最优

“贪心法”这个名字听起来好像不怎么高尚，不过它却是一种常用且实用的思维方式。无独有偶，这种方式也是人类本身所拥有的。

不管是国家还是个人，在制定自己的发展规划时要做到“从一而终”通常非常困难。就拿一个人来说，有句话叫“计划赶不上变化”——在 10 岁的时候规划好的理想人生，可能刚过两三年就会因为很多内在和外在的因素发生了变化而不得不重新规划。可以毫不夸张地说，规划的时间跨度越长就越不实用，因为人生中无处不在的变数会潜移默化地对一个人的规划产生影响，这一系列的变化慢慢积累，到最后可能连规划者本人都看不出当前的情况和原来的规划还有什么联系。

对“帮助计算机解题”来做规划，也会有解决类似问题的困惑，要不怎么说“数据科学贯穿在很多领域之中”呢？而且，对计算机来说，时间和空间成本同样是比较有限的。在一个全局的复杂问题中，如果无法通过穷举、分治、回溯这些问题快速找到一个可行的解决方案，该怎么办？——用贪心法。

贪心法的思路也很明确，就是在每一步向前试探的时候都找到当前的“最优解”，其他的解（分支）一概不看——在有限的视野寻找最优解作为行动纲领。经典算法中的迪杰斯特拉算法（Dijkstra Algorithm）就是其中之一。

迪杰斯特拉算法用于求解有向图中的最短路径问题。在生产实践中，其应用场景通常是那些需要快速应答且穷举和回溯成本极高的场合。在学术方面，所有人都承认以迪杰斯特拉算法为代表的贪心法求出的通常是次优解，即每次找到的与当前访问位置连接成本最低的点，都只是在这一无法回溯的情况下对找到“最优解”的单方向试探，当然不排除一些被忽略的访问路径是最优解的情况（如图 11–15 所示）。每个有向边所标注的权重数字相当于在这个边上前进所发生的成本，当然是从节点 A 走到节点 H 时的成本越低越好，而用贪心法在这里求出的解显然不是全局的最优解。

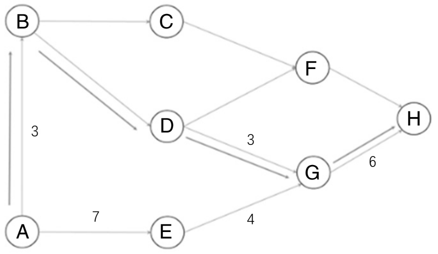


图 11–15 贪心法求解路径问题

不过不用担心，在生产实践中这种次优解已经能够解决实际问题，在求解最优解的成本远远高于求解次优解的情况下就可以大胆使用它。这种“次优解即最优解”的思维方式几乎贯穿于所有工业应用领域，后面我们还会介绍很多这样的例子。

11.7 迭代法——步步逼近

所谓“迭代法”同样不是一个算法，而是一类算法的解题思路。这种方法用来处理那些无法通过解析解的表达得到精确解的情况。

我们都知道，一元二次方程 $y = ax^2 + bx + c$ 的求根公式是

$$y = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

用“配方法”很快就能获解。

对于一元四次及以下的方程，都可以通过求根公式来得到解析解，带入系数的具体值，即可得到确定的数值解。

但是，高次方程及有复杂项的方程就没有这种求根公式[95]，而且很多方程的解都无法直接求出。这使我们不得不去寻求其他方法，迭代法就是其中之一。

95 1831 年，法国著名数学家埃瓦里斯特・伽罗瓦（Évariste Galois，1811 年 10 月 ~ 1832 年 5 月）证明，一元五次以上的方程没有求根公式。

11.7.1 牛顿法

迭代法的核心思路就是用步步逼近的方式来接近理论上的精确值，只要发现当前的试探值已经收敛到一个满足场景要求的误差精度，就可以判断迭代结束，并将这个试探值作为求解的目标值。这种方法可以使很多无法直接求解的问题得到一个足够精确的近似解。这同样也是一种以有限成本的“次优”取代无限成本的“最优”的哲学思想。

迭代法中有一个经典的方法叫“牛顿迭代法”（Newton's Method，或称“牛顿法”），它是牛顿在 17 世纪提出的一种在实数域和复数域上近似求解方程的方法。

例如，有一个一元方程

$$f(x)=0$$

设 r 是满足 $f(x)=0$ 的根。

设置一个初始值 x_0，带入函数 $y=f(x)$，则平面直角坐标系上会有点

$$(x_0, f(x_0))$$

在曲线 $y=f(x)$ 上。

过点 $(x_0, f(x_0))$ 作 $y=f(x)$ 的切线 L_0，L_0 的方程就应该是

$$y=f(x_0)+f'(x_0)(x-x_0)$$

其中，$f'(x)$ 就是 $f(x)$ 的一阶导数。

求出直线 L_0 与 X 轴的交点的横坐标

$$x_1=x_0-\frac{f(x_0)}{f'(x_0)}$$

得到的 x_1 为 r 的一次近似点。

然后，在曲线 $y=f(x)$ 上以相同的方式过点 $(x_1, f(x_1))$ 作 $y=f(x)$ 的切线 L_1，得到 L_1 与 X 轴的交点横坐标

$$x_2=x_1-\frac{f(x_1)}{f'(x_1)}$$

得到的 x_2 为 r 的二次近似点。

以此种方式进行迭代，通项表达式为

$$x_n=x_{n-1}-\frac{f(x_{n-1})}{f'(x_{n-1})}$$

x_n 就成为 r 的 n 次近似点。这个公式称为“牛顿迭代公式”（如图 11-16 所示）。

以上介绍的就是使用一次次迭代来逼近最优解（局部最优解）的过程。

牛顿迭代公式可以推广到高维使用，例如二维、三维且各维度可导的情况。在使用计算机求解的过程中，即使 $f(x)$ 的求导过程非常复杂，也还是有变通的方法。$f(x)$ 的导数 $f'(x)$，不是

用$\lim\limits_{\Delta \to 0}\frac{f(x+\Delta)-f(x)}{\Delta}$来推导解析解，而是根据这个定义来近似求得一个导数值，即某一点的斜率

$$k=\frac{f(x+\Delta)-f(x)}{\Delta}$$

其中，x 可以是函数 $f(x)$ 中任意有定义且可导的值，Δ则取一个足够小的值。

这种思维方式在梯度下降法中也有较好的应用。

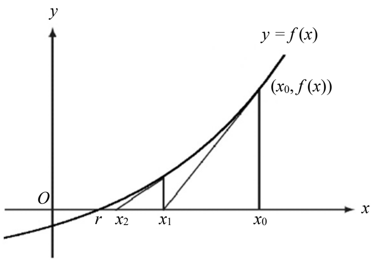


图 11–16　牛顿法迭代中 x 的移动

11.7.2　梯度下降法

在工业领域，还有一些迭代法可以实现这种逼近性质的迭代，例如梯度下降法（Gradient Descent）、随机梯度下降法（Stochastic Gradient Descent）也是用来解决凸优化问题的通用方法。在一维定义域上满足

$$f(ax_1+bx_2)\le af(x_1)+bf(x_2)$$

的就是凸函数。

例如，$y=f(x)=x^2+2$ 就是典型的一元凸函数（如图 11–17 所示），$z=f(x,y)=x^2+y^2$ 则是二元凸函数（如图 11–18 所示），三元及以上的高维凸函数也是有的，只是没法画出图像而已。

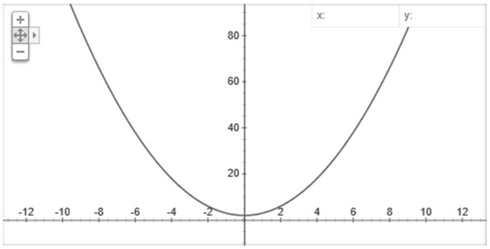


图 11–17　$y=f(x)=x^2+2$ 图像

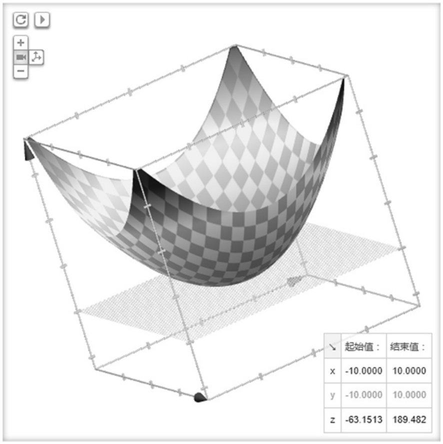


图 11–18　$z=f(x,y)=x^2+y^2$ 图像

在一个问题的求解过程中，如果能把这个问题转化成一个在凸函数上求极值的问题，就算是获解了。在这种情况下需要使用梯度下降法来求极值，核心思路是在函数的曲线（曲面）上初始化一个点，然后让它沿着梯度下降的方向移动移动到函数值极值的位置。这个位置视具体的问题而定，可能是极小值，也可能是极大值（如果是凹函数，那就是梯度上升的方向）。

假设要在一个 n 维空间的凸函数上进行迭代，v 是一个 n 维向量 $v\,(x_1, x_2,\cdots, x_n)$，那么在梯度下降迭代的过程中可以使用这样的原则：

$$(x_i)_{t+1} = (x_i)_t - \eta\frac{\partial f(v)}{\partial x_i}$$

η 是“步长”系数，表示每次迭代移动的距离基数。$\frac{\partial f(v)}{\partial x_i}$ 是在某个 x_i 维度上的偏导数。在连续可导的函数曲面上，任意一次迭代中每个 x_i 都可以更新。在很多场景中，如果能使多元函数在任何一个维度上都是凸函数，通常使用梯度下降法比较容易解决问题（神经网络中的损失函数就是这样）。可以通过定义拟合的残差为关于待定系数的凸函数，用梯度下降法进行优化，从而获解（11.9.3 节会进行详细的讨论）。

11.7.3　遗传算法

不管是低维函数还是高维函数，都可能出现“多峰”的情况，就是在函数图像上出现很多“褶皱”。这种情况下不容易用一般的梯度下降法求解极值问题。因为任意一个初始点都会收敛到最近的一个极小值的位置，而这个极小值有很大概率不是要求的最优解，甚至距离最优解的位置非常远（如图 11–19 和图 11–20 所示）。

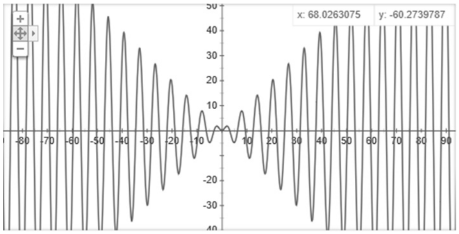


图 11–19　$y = x\sin(x)$ 图像

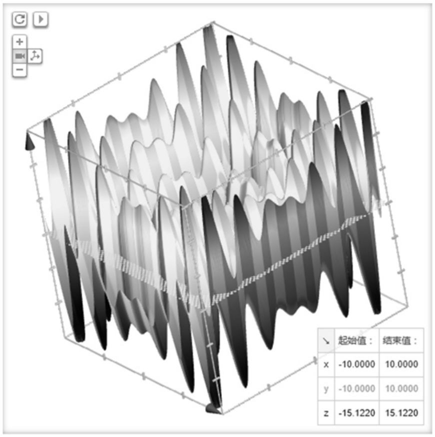


图 11–20　$z = x\sin 2x + y\cos 2y$ 图像

遗传算法作为梯度下降法的一种增强形式，可以以比较高的效率解决高维空间上的极小值查找问题，以及离散数学中的 NP 问题[96]。它的思路就是：在这个候选解出现的高维平面上撒下大量的“种子”，比较这些“种子”的评价函数值，淘汰较小的“种子”；通过基因交叉的方式产生下一代“种子”，比较各代“种子”的评价函数值，再淘汰较小的“种子”；如此反复，直

96 “NP 问题”的全称是“Non-Deterministic Polynomial Complete Problems”，即多项式复杂程度的非确定性问题。

到连续几代出现的“种子”所具备的最大评价函数值不再增加，即可考虑判断收敛。这同样是迭代法的应用。

在解决这种有限定义域内高维多峰函数的极值问题时，使用遗传算法就比较合适，前提是这个多峰函数在任何一个维度内都是连续的。

以函数 $z = x\sin 2x + y\cos 2y$ 为例，$x \in [-10,10]$，$y \in [-10,10]$。

第 1 步：基因编码

这个环节以满足当前场景的精度要求为宜。如果精度为 10^{-5}，那么用一个 20 位的二进制数字作为基因编码即可。因为 $2^{21} = 2097152$，其能够允许的精度已经满足了 −10.00000 到 10.00000 之间 20000001 个最小单位的需求。

设计一个映射函数

$$H(x) = -10.00000 + F(x)\cdot(0.00001)\text{，}\quad F(x) \le 2000000$$

x 是 21 位的二进制编码，例如 $(101010101010101010101)_2 = 1398101$。$F(x)$ 是二进制编码到十进制实数的映射函数。

第 2 步：设计初始群体

设计一定量的群体，每个个体都用一个二进制编码来表示，使用程序循环生成 N 个 21 位二进制编码，当发现 $F(x) > 2000000$ 时就进行舍弃并重新生成，直到生成 N 个满足要求的 21 位二进制编码。

第 3 步：遴选与生育

首先，比较当前 N 个 $H(x)$ 的值。然后，将其从大到小排序，选择最大的前 M 个对象作为可以生育后代的备选。最后，在这 M 个备选对象中进行两两之间的基因重组（如图 11–21 所示）。

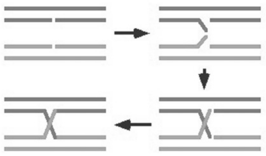


图 11–21　基因重组

根据排列组合，每一代会产生 $2C_M^N = 2\cdot\dfrac{M!}{N!(N-M)!}$ 个下一代对象。在两两组合的过程中，可能会在这 21 位中的第 1 位后到第 21 位前的任意位置断开然后重组，这个位置是随机的。

以 2 个基因组为例：

101010101010101010101
000011110000111100001

假如在第 7 位后断开，那么生成的 2 个后代的基因组就是

000011101010101010101
101010110000111100001

最终，一共产生了$2\cdot\frac{M!}{N!(N-M)!}$个新的后代基因。

在当前的 M 个评价函数值最大的基因和$2\cdot\frac{M!}{N!(N-M)!}$个新的后代基因中再次进行这$2\cdot\frac{M!}{N!(N-M)!}+M$个评价函数的排序，并选择前 M 个作为产生下一代的备选基因。

在这个过程中，可以用概率 p 来调整其中某一位由 0 变成 1 或由 1 变成 0 的可能性，这个环节叫作“变异”，它给某个基因组提供了从一个多维空间中评价函数较低位置跳跃到较高位置的机会。虽然也有一定的概率会从较高位置跳跃到较低位置，但是跳跃到较高位置的会有更大的概率被保留下来。

接下来，循环进行这个遴选与剩余的过程，直到连续 K 轮的函数最大值不再增加，就可以判断为函数收敛。在这个算法中，N、M、K 都属于可调整的参数。一般来说，N、M、K 越大，找到优质解的可能性也就越大（代价是消耗的计算资源增多）。

遗传算法是启发式（Heuristic Algorithm）算法的一种，因为在这种几乎没有确切方向的优化场景中，需要让计算机来进行一定的“智能”选择。这种“智能”就是我们通过对基因的编码，以及对基因编码的交换，从中找出那些在一轮一轮的遴选中由计算机挑选出来的编码体现出更好特性的对象。这些优秀的对象所蕴含的编码就是逼近最优解的模板，并通过不断尝试重构这些编码来获取更好的解，这一发掘过程是一种人启发计算机的过程。

梯度下降法也好，牛顿法也罢，得到的都是一个近似解，都不是理想的、满足条件的最优解——总是差那么一丁点。但是，只要这个差距足够小、能够满足工程需要就够了，毕竟即使求出了这个最理想的位置，在实现中也会由于各种其他误差的引入而使为向这个理想位置逼近的努力付诸东流。这就是理想和现实的区别，希望这个方法能够治好大多数人的“强迫症”。

在生产实践中解决，我们通常会想办法把一个复杂问题的评价函数变成一个凸函数或者连续函数，然后通过迭代法逐步逼近，使问题得到解决。只要算法设计合理，往往能起到事半功倍的效果。

11.8　机器学习——自动归纳

谈到机器学习，就要先说说数据挖掘。数据挖掘是随着商务智能发展起来的一种相对比较新的一种算法学科，而且数据挖掘与机器学习有很多相互覆盖的子领域或者算法，例如贝叶斯概率、聚类、分类、回归等概念在机器学习和数据挖掘领域都有讨论。

要进行数据挖掘，就要有一定量的数据作为研究对象。“挖掘”这个词很形象，说明一些东西并不是在表面，让人一眼就能看明白，而是要进行深度的研究、对比、甄别等工作，最终从中

找到规律或者获得知识。在我看来，数据挖掘和机器学习研究的都是数据在深层的联系问题，并进行了相应的体系性量化工作。

机器学习也是近年来非常火爆的一个计算机算法分支学科。以前的算法大都是为了解决一个模型上确定的问题，由人直接编写，由计算机根据算法的指示一步步执行顺序、循环、分支这样的流程逻辑来处理输入的数据。而机器学习要解决的问题是，通过设计算法，对输入的数据进行归纳，然后根据归纳的结果进行相应的响应输出，这是二者最大的区别。这个归纳的目标就是条件尽可能准确，这也是评价一个训练算法是否优秀的重要条件。这种方式很像人类学习客观事物的过程，所以得名“机器学习”。

机器学习可以粗略地分成非监督学习（Non-Supervised Learning）、监督学习（Supervised Learning）、半监督学习（Semi-Supervised Learning）及强化学习（Reinforcement Learning）等多个大类。

11.8.1 非监督学习

非监督学习直接对输入的数据集建模，例如聚类（Clustering）和隐马尔可夫模型（Hidden Markov Model，HMM）等。

非监督学习，顾名思义，是一个不加干预的学习过程。在这个学习过程中，每个被观察的对象没有明确的标签属性，只有量化特征维度 X。X 是一个 n 维向量，分量分别是

$$(x_1, x_2, \cdots, x_n)$$

然后，在这个 n 维空间里尝试寻找聚类（如图 11-22 所示）。

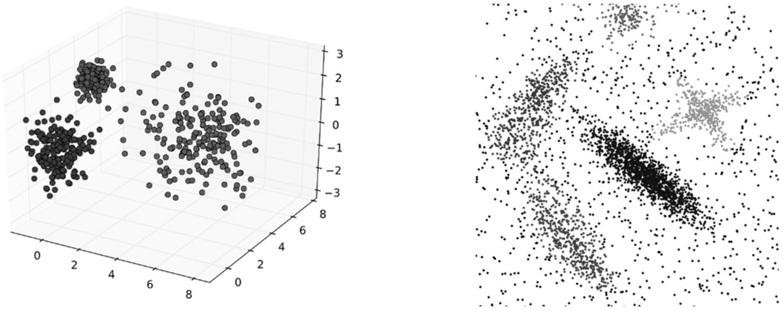


图 11-22　三维空间和二维空间聚类效果

聚类研究的对象是由距离或者稠密程度形成的聚簇对象及其特点。如图 11-22 左图所示就是一个三维空间 K-Means 算法聚类的效果图。K-Means 算法的步骤大致如下。

（1）从 n 个向量对象中任意选择 k 个向量作为初始聚类中心。

（2）根据在步骤（1）中设置的 k 个向量（中心对象向量），分别计算每个对象与这 k 个向量的距离。这里的距离都是欧氏距离（Euclidean Distance）。

（3）对于步骤（2）中的计算，任何一个向量与向量 k 之间都有一段距离，有的远，有的近。把这个向量和距离它最近的中心向量对象归在一个类簇中。

（4）重新计算每个类簇的中心对象向量位置。

（5）重复步骤（3）、（4），直到类簇聚类方案中的向量归类变化极小为止。

由于 K-Means 算法是给定若干个聚类中心点的向量，通过计算空间重心的方式进行聚合，所以聚合出来的基本都是某一半径的高维“球体”包络下的空间向量集合。

而对如图 11-22 右图所示的聚簇形状不规则的数据集，应使用密度聚类的方式（例如 DBSCAN 算法）。除此之外，就是研究那些离群点（也叫“孤立点”，是指那些不在任何聚类当中的离散的点向量）。对孤立点的研究，通常也是研究它的一些特性或者成因。

另一种典型的非监督学习叫作“隐马尔可夫模型”，也称“马尔可夫链”（如图 11-23 所示）。马尔可夫链是一个数学概念，因为它由俄罗斯物理学家、数学家安德烈・马尔可夫（A. A. Markov）提出的，所以得名。马尔可夫链的核心是：在给定当前知识或信息的情况下，观察对象过去的历史状态对于预测将来而言是无关的。也可以说，在观察一个系统中序列值变化的时候，下一个状态（第 $n+1$ 个状态）如何的概率只需要观察和统计当前状态（第 n 个状态）即可正确得出。另外，我们在一些资料中会看到贝叶斯信念网络的分类模型概念。隐马尔可夫链和贝叶斯信念网络的模型，其思维方式有些相似，区别在于隐马尔可夫链的模型更为简化，或者我们可以认为，隐马尔可夫链就是贝叶斯信念网络的一种特例。而且，隐马尔科夫链是一个双重随机过程，不仅状态转移之间是一个随机事件，状态和输出之间也是一个随机过程。

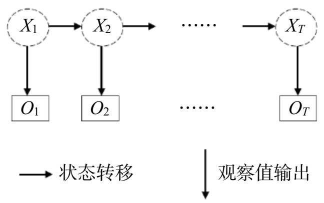


图 11-23　马尔可夫链

在一个完整的观察过程中有一些状态的转换，就是如图 11-23 所示用虚线圈标出的 X_1 到 X_T。在观察中，X_1 到 X_T 的状态存在一个客观的转化规律，但是我们无法直接观测到。我们观测到的是每个 X 状态下能让我们看到的输出 O，也就是 O_1 到 O_T 这些输出值。我们需要通过这些输出值进行模型建立和状态转移的概率计算。

这种学习过程的结果是得到一个概率转化矩阵。这是一个无监督的过程，不需要人对它进行任何标注。在得到转化矩阵之后，能做的事情就比较多了，基本都是关于序列预测的。这种模型衍生出了多种重要的算法，例如维特比算法（Viterbi Algorithm）、前向算法（Forward Algorithm）和后向算法（Backward Algorithm）。

我们可以这样看待隐马尔可夫模型这种非监督学习方式。

在一个长期产生输出的序列中，我们能够观察到的输出的序列对象就是 O_1 到 O_T。每一次输出的序列都用一个向量表示，这个向量也同样能够形成一个集合 Q，每个成员用 Q_i 表示，i

是从 1 到 T 的正整数。通过对一个邻接矩阵的统计，可以得到一个转换的概率关系（如表 11–6 所示）。

表 11–6 转换概率矩阵

	Q_1	Q_2	……	Q_T
Q_1	P_{11}	P_{12}	……	P_{1T}
Q_2	P_{21}	P_{22}	……	P_{2T}
……	……	……	……	……
Q_T	P_{T1}	P_{T2}	……	P_{TT}

这个概率矩阵通过对任意 Q_m 到 Q_n 项的转换数量的统计来计算转换概率，而且在这个模型中，在统计环节也只看输出项 O_i 和 O_{i-1} 的关系。对建立一个认知模型来说，过程已经得到了极度简化，而且我们也不用担心 O_i 与 O_{i-2} 的关系没有被计入。在隐马尔可夫模型的应用中，对 X_i 和 O_i 的划分是比较宽泛的。如果我们觉得讨论前后的概率关系需要参考一些更远的状态，只要将临近的多个 X_i 和 O_i 合并观察和建模即可。观察状态的时间宽度在隐马尔可夫模型中没有严格的限制，可以是实际生产中的 1 秒、1 分钟、1 小时、1 天或更久，所以 O_i 与其前部很远位置的状态其实也会根据观察者观察尺度的变化而变化。

这种认知方式对前后变化规律明显的模型有极大的应用优势。例如，隐马尔可夫模型在 NLP（Natural Language Processing，自然语言处理）领域有着得天独厚的优势，在不同的场景中，语音所表示的语义前后相关性极高，所以同音汉字在这种前后相关的语境中可以通过统计学及隐马尔可夫模型实现最大概率汉字的检出。除此之外，通信工程领域 3G/4G 广泛使用的码分多址（Code Division Multiple Access，CDMA）技术也是典型的以隐马尔可夫模型为理论基础的工学应用。

11.8.2 监督学习

监督学习和非监督学习的方式有所不同。在监督学习的范畴，样本对象除了有自己的特征向量以外，还有标签向量（或称“分类向量”）。以特征向量表示输入，以标签向量表示输出或预测值，为确立这样的映射关系让计算机进行的自动化归纳运算就是监督学习。

监督学习有两个工作环节，一个是“学习”，另一个是“分类”。

通过大量训练样本的特征向量及它们各自的分类向量，让计算机根据算法逻辑对这个映射模型进行归纳，这个过程就是“学习”过程。用这个学习过程得到的结果模型去对任意一个输入的特征向量进行分类判断，就是“分类”过程。

这种“学习”和“分类”的过程是绝大多数监督学习的工作模式。在实际应用中，“学习”和“分类”迭代进行，用“分类”的结果验证“学习”的正确性，调整“学习”后，再用不断

的“分类”实践来进行验证和反馈调整——这就是监督学习的完整过程。

监督学习的体系非常丰富，看上去让人有点眼花缭乱，但是仔细品味一下，做的都是同一件事情。

1. 贝叶斯概率

有一个著名的公式

$$P(A\mid B)P(B)=P(B\mid A)P(A)$$

或写作

$$P(A\mid B)=\frac{P(B\mid A)P(A)}{P(B)}$$

这就是朴素贝叶斯模型公式（Naive Bayesian，如图 11–24 所示）。其中，$P(A)$ 叫作事件 A 的先验概率，是指一般情况下我们认为事件 A 发生的概率；$P(B\mid A)$ 叫作似然度，是指事件 A 假设条件成立的情况下发生事件 B 的概率；$P(A\mid B)$ 叫作后验概率，是指在事件 B 发生的情况下发生事件 A 的概率，也就是我们要计算的概率；$P(B)$ 叫作标准化常量，和事件 A 的先验概率定义类似，就是一般情况下事件 B 发生的概率。

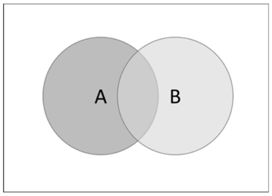


图 11–24　朴素贝叶斯（1）

贝叶斯决策理论方法是统计模型决策中的一个基本方法，其基本思想如下。

（1）已知类条件概率密度参数表达式和先验概率。

（2）利用贝叶斯公式将其转换成后验概率。

（3）根据后验概率的大小进行决策分类。

朴素贝叶斯分类的方式则不太一样。贝叶斯概率研究的是条件概率，也就是说，研究的场景就是带有某些前提条件或者在某些背景条件的约束下发生的概率问题。

上面基于对于样本空间中两个事件（事件 A 和事件 B）的条件概率描述，完整的贝叶斯公式如下。

如图 11–25 所示，设 $D_1, D_2, \cdots, D_n$ 为样本空间 S 的一个划分，如果以 $P(D_i)$ 表示 D_i 发生的概率，且 $P(D_i)>0$ $(i=1, 2, \cdots, n)$。对于任一事件 x，$P(x)>0$，有

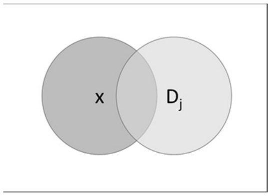


图 11–25　朴素贝叶斯（2）

$$P(D_j\mid x)=\frac{P(x\mid D_j)P(D_j)}{\sum_{i=1}^{n}P(x\mid D_i)P(D_i)}$$

也就是说，这可以推广到事件 x 和事件集合 D_j。

D_i 表示不同的事件划分，而且用 D_i 可以把整个空间划分完毕。在每个 D_i 事件发生的同时

都进行事件 x 发生与否的记录，并记录在 D_i 发生的情况下 x 的发生概率。

$\sum_{i=1}^{n} P(x|D_i)P(D_i)$ 可以化简成 $P(x)$，因为它代表整个事件空间内在所有的事件 D_i 发生的前提下发生事件 x 的概率的一个加和。

所以，最后等式两边就化简成

$$P(D_j|x)P(x) = P(x|D_j)P(D_j)$$

也就是说，在全样本空间内，发生 x 的概率乘以在发生 x 的情况下发生 D_j 的概率，等于发生 D_j 的概率乘以在发生 D_j 的情况下发生 x 的概率。

如图 11–25 所示，左侧的圆面积代表发生 x 的概率，右侧的圆面积代表发生 D_j 的概率，中间的交集就是等号两边各自表示的内容，也就是 $P(D_j|x)P(x)$ 或 $P(x|D_j)P(D_j)$ 的值。

朴素贝叶斯分类器是在机器学习中应用最为广泛的一种分类器。与其说它是一个公式，不如说它是一种思想或者思维方式。对贝叶斯概率解决的问题我们可以这样看待：对任何事情的认知，首先可以通过统计的方法得出事情发生的概率。这个概率叫作“先验概率”，是指在一种非特定（或不明确）条件下的概率判断。然而，一旦引入了其他维度（也就是事件认知维度增加），如果通过统计发现其中一件事情的增加或减少会影响另一件事情发生的概率，那么这两件事情的关联性可以通过统计量化表示。这个量化表示非常重要，它不仅能够说明一件事情的发生对另一件事情的发生概率有影响，还能告诉我们这个影响具体有多大，也就是说，贝叶斯概率模型是量化因果关系的科学认知方法。

2. 回归

回归（Regression）也是一种通过归纳样本特征向量和分类向量的关系得到模型表达式的过程。常见的回归有线性回归（Linear Regression）、非线性回归（Non–Linear Regression）和逻辑回归（Logistic Regression）等。

线性回归的表达式是

$$f(x) = w^T x + b$$

其中，x 是一个 n 维向量，或者可以看成一个 $n\times1$ 的矩阵，w^T 是一个 $1\times n$ 的矩阵，两者作内积再加 b，就是一个一维向量（一个实数值）。

$f(x) = w^T x + b$ 最常见、最普及的形式是 $f(x) = ax + b$。这种形式我们每个人在上中学的时候都见过，也是一维向量 x 的特例形式。在这种情况下做拟合，就是把每个向量 x 和它的分类向量 y 带入分类器中进行训练，最终推导出最为合适的 w^T 和 b。

“最为合适”是什么概念呢？在回归模型的认知体系中，通常把能够保证全局误差小的 w^T 和 b 作为要求的解，误差越小，说明这种 w^T 和 b 对描述模型来说越准确。以 $f(x) = ax + b$ 为例，可以把这个求解最优化 a 和 b 的问题变成一个最小化损失函数 $Loss$（误差函数）的问题。

对机器学习中的训练来说，损失函数 *Loss* 的设计也有一些讲究。

第一，它应该具有相对明确的统计学或物理解释，能够比较直观地说明这个值的大小和误差之间的量化关系，并且恒为非负数。

第二，损失函数 *Loss* 是否能够通过相对容易的方式进行优化。如果能够以我们熟悉的方式进行优化，就比较好。此外，损失函数 *Loss* 的优化过程收敛速度要快。

常见的损失函数有绝对值损失函数（Absolute Loss Function）、平方损失函数（Squadratic Loss Function）、0–1 损失函数（0–1 Loss Function）、对数损失函数（Logarithmic Loss Function）和 Hinge 损失函数（Hinge Loss Function）。除此之外，还有一些损失函数的构造方法。在不同的场景中，损失函数通常不建议通用，原则上还是参考刚刚提出的两点。涉及哪个损失函数，就解析哪个损失函数。

绝对值损失函数的表达式为

$$Loss(Y, f(X)) = \sum_{i=1}^{n} |Y_i - f(X_i)|$$

就是将每个分类向量 Y 和拟合值 $f(X)$ 的差的绝对值进行加和运算，其物理解释非常明确，非常适合定义域和值域都是实数的空间向量的线性拟合模型。这种损失函数的优化并不困难，而且不会出现梯度越来越小时收敛速度越来越慢的问题，所以很多工业应用场合喜欢使用它。

平方损失函数的表达式为

$$Loss(Y, f(X)) = \sum_{i=1}^{n} (Y_i - f(X_i))^2$$

这个形式与绝对值损失函数有些相近，区别在于不是对差取绝对值进行加和运算，而是对差值的平方进行加和运算。从物理意义上看，这种方式与绝对值损失函数区别不大，而且求出的待定系数在这两种情况下互为充要条件。但是，平方损失函数的优势很明显，它是一个处处可导的凸函数，可以用梯度下降法找到极小值。这是在假设样本是高斯分布的条件下得到的结论。

这种用平方损失函数的方式求解线性回归拟合中的待定系数的方法也叫作“最小二乘法”（Least Squares Method），在这个例子中写作

$$Loss(a,b) = \sum_{i=1}^{n} (y_i - (ax_i + b))^2 \quad \Leftrightarrow \quad Loss(a,b) = \sum_{i=1}^{n} [y_i^{\,2} + a^2 x_i^{\,2} + 2abx_i + b^2 - 2y_i(ax_i + b)]$$

现在，这个问题就转化成了使用梯度下降法求解凸函数 $Loss(a, b)$ 极小值的问题。如果希望得到解析解，在这种二维 (a, b) 的情况下还是可以实现的。而对于 w^T 维度极高的情况，就最好使用梯度下降法求解了。

线性回归中使用的这种思维方式就是把拟合过程变成一个关于 $Loss(w^T)$ 的凸优化问题，其原则就是使全局的残差值最小化。这种方式是一种应用非常普遍的拟合方式。

对于非线性回归，通常有两种思路，其中一种是先把它转化成线性回归的模型进行拟合，其余不能转化的则需要通过经验、观察、多次尝试等手段进行拟合，并从中找出拟合误差最小的情况。对于拟合残差的问题，思路和线性回归一般无二，在这里我们就不用过多的篇幅进行描述了，只简单介绍一些常见的非线性拟合的模型曲线。

双曲线（如图 11–26 所示）：$\frac{1}{y}=a+\frac{b}{x}$

二次曲线（如图 11–27 所示）：$y=ax^2+bx+c$

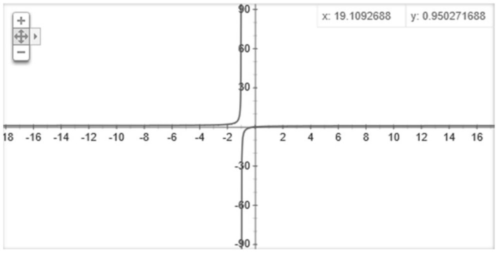


图 11–26 双曲线 $\frac{1}{y}=1+\frac{1}{x}$

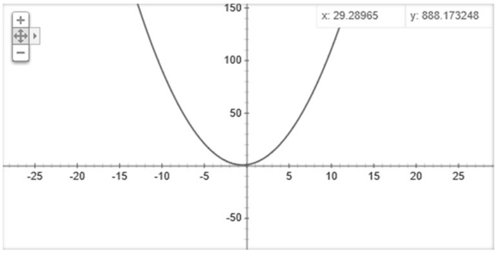


图 11–27 二次曲线 $y=x^2+x+1$

三次曲线（如图 11–28 所示）：$y=ax^3+bx^2+cx+d$

幂函数曲线（如图 11–29 所示）：$y=ax^{\alpha}$

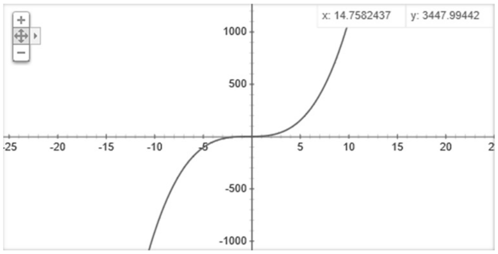


图 11–28 三次曲线 $y=x^3+x^2+x+1$

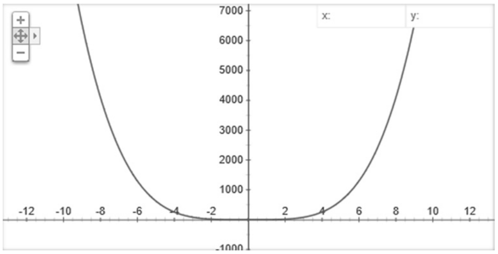


图 11–29 幂函数曲线 $y=x^4$

指数函数曲线（如图 11–30 所示）：$y=a^x$

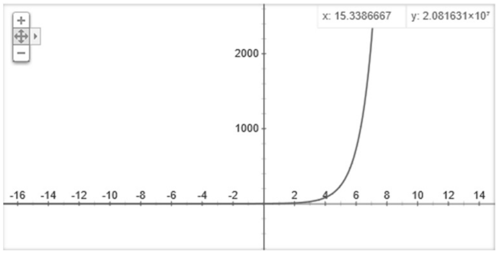


图 11–30 指数函数曲线 $y=3^x$

对数函数曲线（如图 11–31 所示）：$y = a + b\log x$

S 曲线函数（如图 11–32 所示）：$y = \dfrac{1}{a + be^{-\alpha x}}$

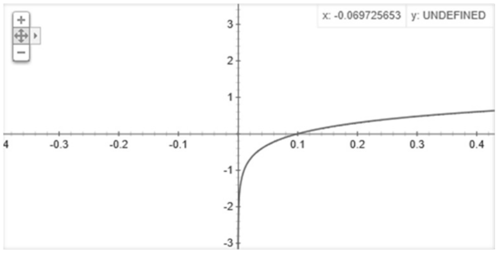


图 11–31　对数函数曲线 $y = 1 + \lg x$

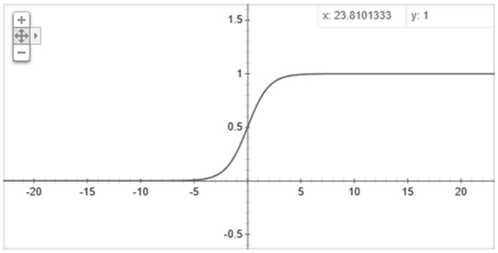


图 11–32　Sigmoid（S 曲线）函数 $y = \dfrac{1}{1 + e^{-x}}$

关于非线性回归，我们只讨论一下逻辑回归，这种方式在机器学习中的应用还是很广泛的。

逻辑回归和普通线性回归的区别在于，普通线性回归的分类标签通常是连续的实数，而逻辑回归是离散的分类标签 0 和 1。

逻辑回归的函数表达式为

$$f(t) = \frac{1}{1 + e^{-t}}, \quad t = w^T x + b$$

这个函数其实是由两个函数组合而成的，一个是 $t = w^T x + b$，另一个是一个线性函数。其中，$f(t) = \dfrac{1}{1 + e^{-t}}$ 是一条 S 形曲线，$t = w^T x + b$ 是一个典型的线性回归分类器模型，在前面已经讨论过了。通过 $f(t) = \dfrac{1}{1 + e^{-t}}$ 的映射，在 y 的值域上形成了一个值域为 0 和 1 的伯努利分布。

所谓伯努利分布就是一个只有 0 和 1 的分布，可以写作

$$P_n = \begin{cases} p & n = 1 \\ 1 - p & n = 0 \end{cases}$$

也可以写作

$$P(n) = p^n (1 - p)^{1-n}$$

在逻辑回归中，可以考虑使用平方损失函数，只不过平方损失函数和逻辑回归本身的函数分布目的不同。逻辑回归希望得到的是伯努利分布的形式，也就是希望得到一个概率。

通过代换

$$
\begin{aligned}
& t = \ln(\frac{p}{1-p}) \\
& \Leftrightarrow e^t = \frac{p}{1-p} \\
& \Leftrightarrow e^t(1-p) = p \\
& \Leftrightarrow e^t = p + e^t p \\
& \Leftrightarrow p = \frac{1}{1+e^{-t}}
\end{aligned}
$$

p 是一个 (0, 1) 之间的实数，而且 $p=\frac{1}{1+e^{-t}}$ 的曲线是连续可导的。

损失函数对单个样本的损失值是

$$\text{Cos}\,t = -\log(y)\,,\quad y=1$$

$$\text{Cos}\,t = -\log(1-y)\,,\quad y=0$$

所以，总的损失函数为

$$Loss = \sum_{i=1}^{n}\text{Cos}\,t_i$$

$$Loss = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot \log f(x_i) + (1-y_i)\cdot\log(1-f(x_i))$$

展开后，前后两项的 $-\log(x)$ 在定义域上是凸函数，所以这个问题又转换成了凸优化问题。

在回归这种建模理念中，同样非常重视模型描述的简洁性，所以在应用中直接使用非线性规划曲线来进行非线性回归远没有使用线性回归普及，而且非线性回归对高维空间的模型拟合难度非常大，其原因主要也是这种拟合后对以残差减小为目的的优化过程难以进行。

3. 决策树

决策树（Decision Tree，如表 11-7 所示）也是一种相对比较常见的经典分类学习算法，面对的训练对象同样是一些输入向量 X_i。X_i 是一个 n 维向量及 X_i 所对应的分类标签。

表 11-7　决策树

维度 1	维度 2	……	维度 N	分类
X_{11}	X_{12}	……	X_{1n}	Y_1
X_{21}	X_{22}	……	X_{2n}	Y_2
……	……	……	……	……
……	……	……	……	……
……	……	……	……	……
X_{n1}	X_{n2}	……	X_{nn}	Y_n
……	……	……	……	……

对这类问题，决策树算法都希望用尽可能简洁并准确的方式描述分类过程，并平衡“简洁”和“准确”这对矛盾。

在这个算法中，需要对整个训练样本集分类进行熵的评估，其目的是衡量这种分类本身的杂乱程度。这里的“熵”和《信息论》中所说的“熵”是同一个概念。从公式上来看，分类的数量越多，发生的概率越均等，熵的值就越大。根据《信息论》的经典理论——信息是用于消除不确定性的东西，那么在这样一个环境中，如何通过尽可能短的“消息”来尽可能多地消除不确定性，就成了破解这类问题的主要思路，而这个所谓的“消息”就是构造出来的分类树的“树枝”（如图 11–33 所示）。

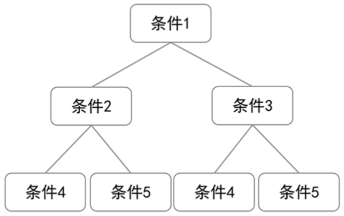


图 11–33　决策树的构造

一般来说，通过一个维度条件的约束就能确定分类是不是最好的。也就是说，如果只用“条件 1”这个节点来划分，给予一个维度的约束条件，就已经能把整个分类描述“精确”，那就完全没有必要加入其他任何维度的描述了（因为那些维度不能带来消除不确定性的信息）。

对于构造一层“决策树”来说，引入哪个字段最为合理，完全取决于这个维度带来的信息增益——也就是熵减的大小。熵减越大，就说明这个维度对于分类的划分越有利。在这个环节，经典的算法有 ID3、C4.5 和 CART。

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

ID3 直接对各个维度的信息增益进行计算，选择信息增益最大的维度作为决策点，将集合分成两部分，然后在第二层继续选择剩余维度中信息增益最大的作为决策点，如此一层一层构造下去。随着层数增多，信息熵逐渐减少，这个过程就是求解过程。

$$Info_A(D) = -\sum_{j=1}^{k} \frac{|D_j|}{|D|} \times Info(D_j)\text{，}\quad Gain(A) = Info(D) - Info_A(D)$$

C4.5 和 ID3 的流程看上去没有本质的区别，都是在寻找信息熵逐渐减小的路径。不过，ID3 的算法导致它倾向于使用取值较多的维度。而 C4.5 就没有这个问题，它使用信息增益比来代替信息。

$$Gini_A(D) = 1 - \sum_{i=1}^{c} p_i^2\text{，}\quad \Delta Gini(A) = Gini(D) - Gini_{\mathrm{A}}(D)$$

CART（Classification and Regression Tree）则使用标准的二叉树，在节点裂解的时候使用基尼系数进行计算。基尼系数的含义与信息熵的含义类似，是用来量化纯度的。混杂度越高，基尼系数就越高。通过计算，得到使基尼系数降低幅度最大的维度作为裂解所使用的维度。

最后，通过剪枝法对生成的树模型进行修正，分为“前剪枝”和“后剪枝”两大类方法。

前剪枝是指在应用所有的维度条件之前就终止树的生成的方法，只要目前的树的层数和分类的不纯度都已经逼近业务场景能够接受的极限就可以了。

后剪枝是指在决策树构造完成后对树进行修剪，主要目的是消除过拟合。常见的算法有降低错误剪枝（Reduced Error Pruning，REP）、悲观错误剪枝（Pessimistic Error Pruning，PEP）、基于错误剪枝（Error-Based Pruning，EBP）、代价-复杂度剪枝（Cost-Complexity Pruning，CCP）、最小错误剪枝（Minimum Error Pruning，MEP）等。

这些剪枝方法各有优缺点，总体来说是在降低过拟合程度这个环节上平衡算法复杂性和精确度改善这对矛盾，这里也可能产生过拟合现象。因为只要引入的样本不够多，就有产生过拟合的风险，所以在验证集上进行验证后，强烈建议使用后剪枝方法。

4. 支持向量机

支持向量机（Support Vector Machine，SVM）是机器学习分类模型中适用范围比较广的一种算法。

在用支持向量机求解分类问题的过程中有这样一个视角，就是在多维空间中找到一个平面，使训练集中的两类多维向量与这个平面的距离最大化。而每个向量到平面的距离之和最大化的问题，可以转化为一个等价的凸函数，将其作为损失函数最小化的问题求解——仍然是凸优化问题。

例如，给定一个训练数据集

$$\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$$

x_i 是一个 m 维向量，y_i 是分类标签。为了讨论方便，这里使用 {+1，−1} 来表示其取值。使用支持向量机进行训练的目的实际上是在 m 维空间中找到一个可以分离两类向量的超平面。

这个超平面的定义为方程

$$wx+b=0$$

x 是一个 m 维的向量，或表示成一个 $m\times 1$ 的矩阵。w 是一个 $1\times m$ 的矩阵。b 是一个实数。

分类的决策函数是

$$f(x)=sign(wx+b)$$

也称作“线性可分支持向量机”。

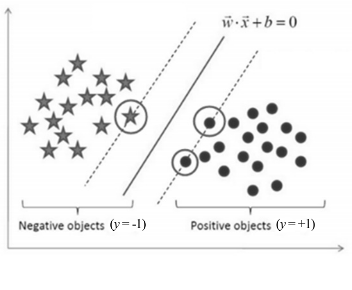


图 11-34 二维空间向量超平面

以二维空间的向量为例，最后希望用如图 11-34 所示的方式找到超平面 $wx+b=0$，其中需要学习的就是 w 和 b。在这里有一个假设，就是在分类正确的情况下距离超平面越远的点其确信程度越高，也就是在超平面方程 $wx+b=0$ 给定的情况下，可以使用一个间隔函数 $d(x)=a|wx+b|$ 来表示某

个向量到超平面的距离。

某个向量点 x_i 到超平面的距离用欧氏距离（Euclidean Metric）表示，可以写作

$$\gamma_{\mathrm{i}} = \left|\frac{w}{\|w\|}x_i + \frac{b}{\|w\|}\right| \quad 或者 \quad \gamma_{\mathrm{i}} = \frac{1}{\|w\|}|wx_i + b|$$

根据 y_i 取值的特点，可以写作

$$\gamma_{\mathrm{i}} = y_i\left(\frac{w}{\|w\|}x_i + \frac{b}{\|w\|}\right)$$

因此也叫作“几何间隔”。其中，$\|w\|$ 叫作“范数”，在欧式空间中的欧式范数定义为

$$\|w\| = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$$

是向量的模。

几何间隔在空间几何上有具体的几何距离解释，这和我们在二维空间中求点到直线的距离及在三维空间中求点到平面的距离的计算方式是一样的。如果在规范化后，令 $\|w\|=1$，间隔距离可以写作

$$\overset{*}{\gamma}_{\mathrm{i}} = y_i(wx_i + b)$$

也叫作“函数间隔”。

定义一个训练集中的样本点到超平面 $wx + b = 0$ 几何距离最小的点

$$\gamma = \min_{i=1,\ldots,n} \gamma_i$$

所以，现在的问题就变成了怎样让 γ 最大化。

支持向量机学习的基本思路是，找到能够正确划分训练数据集且几何间隔最大的分离超平面。对线性可分的训练数据集来说，线性可分的分离超平面有无穷多个，但是几何间隔最大的分离超平面是唯一的。

“间隔最大化”又称“硬间隔最大化”，其直观解释是：对训练数据集找到几何间隔最大的超平面，意味着以充分大的确信度对训练数据进行分类，对最贴近超平面的点也有足够大的确信度将它们分开，以保证获得很好的分类预测能力。

在线性可分的情况下，训练数据集的样本点中与分离超平面距离最近的样本点的实例称为“支持向量”（Support Vector，如图 11-34 所示用圆圈标出的五角星和圆点）。

在 SVM 进行训练的过程中，会使用一种叫作“Hinge Loss”（也译作“合页损失函数”）的函数来充当损失函数。这个函数写作

$$\sum_{i=1}^{n}[1 - y_i(wx_i + b)]_+ + \lambda\|w\|^2$$

$$[z]_+ = \begin{cases} z, & z > 0 \\ 0, & z \le 0 \end{cases}$$

前半部分很容易理解：如果 $y_i(wx_i + b)$ 分类预测与训练时的分类一致，其值就是 1，$1 - y_i(wx_i + b)$ 的值就是 0；相反，如果 $y_i(wx_i + b)$ 分类预测与训练时的分类不一致，其值就是 0，$1 - y_i(wx_i + b)$ 的值就是 1。后半部分 $\lambda \| w \|^2$ 叫作正则化项，λ 是系数，$\| w \|^2$ 叫作 w 的 L_2 范数（$\| w \|^2$ 实际上就是 $\sqrt{\sum_{i=1}^{n} w_i^2}$）。

正则化项在很多机器学习损失函数中都可以看到，这是一种为了降低结构风险而设置的“损失值”。由统计得到的 $1 - y_i(wx_i + b)$ 这个部分的损失称为经验风险，后面的正则化项 $y_i(wx_i + b)$ 称为结构风险。尤其是在这种模型中，w 按比例扩大不会影响分类结果，我们又希望得到一个 w 尽可能小的超平面描述来保证其简洁，且不会由于 w 过大而对噪声敏感进而产生过拟合现象。这种对模型进行优化的思路同样适用于其他分类模型。

合页损失函数的图形如图 11–35 所示，横轴表示间隔函数 $1 - y(wx + b)$，纵轴表示损失函数的值，还有一个用来做对比的 0–1 损失函数。虽然 0–1 损失函数对损失的解释更为直接，但是由于它有不可导的地方，而且可导的地方导数都是 0，所以优化这种损失函数非常困难，这才使用了合页损失函数——从形状上来看，确实像一片打开的合页。合页损失函数在分类正确的时候损失才是 0，否则损失是 $-y_i(wx_i + b)$。所以，合页损失函数的优点是分类要求更高，当确信度足够高的时候损失才会下降为 0。

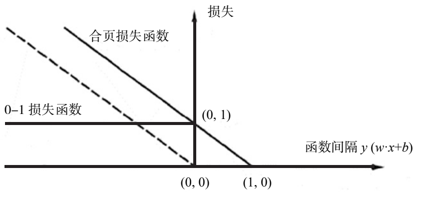


图 11–35 合页损失函数

如果线性不可分，在 SVM 机器学习算法中会使用一种叫作“核函数”（Kernel）的函数对数据进行升维操作，这也是对非线性分类问题的处理。

核函数是一种技巧，通过把当前的 x 向量以一个函数 $K(x, z)$ 映射到高维空间，让 $K(x, z)$ 映射后的可以用于分类的超曲面方程在低维空间中仍然呈现超平面的形态。

$$K(x, z) = \phi(x) \cdot \phi(z)$$

就是一个想定的核函数形态。

将 $K(x, z)$ 定义为函数 $\phi(x)$ 与 $\phi(z)$ 的内积。其最直观的例子就是在二维空间里假设有一个椭圆能够把空间划分为两部分（这是线性不可分的），椭圆的方程为

$$w_1 x^2 + w_2 y^2 + b = 0$$

定义

$$z = \phi(x) = (x^2, y^2)^T$$

则原来的椭圆方程就可以退化为

$$w_1 z_1 + w_2 z_2 + b = 0$$

这就是一个 (z_1, z_2) 坐标系中的直线方程了。

在核函数的技巧中，可以通过不构造 $\phi(x)$ 的方式构造符合要求的核函数 $K(x, z)$。这里有一个充要条件，就是让 $K(x, z)$ 为正定核。设 $\chi \subset R^n$，$K(x, z)$ 是定义在 $\chi \times \chi$ 上的对称函数，如果对任意 $x_i \in \chi$，$i = 1, 2, \cdots, m$，$K(x, z)$ 对应的 Gram 矩阵 $K = [K(x_i, x_j)]_{m\times m}$ 是半正定矩阵，则称 $K(x, z)$ 是正定核。

常用的核函数有多项式核函数

$$K(x, z) = (x \cdot z + 1)^p$$

高斯核函数（径向核函数）

$$K(x, z) = \exp(-\frac{\| x - z \|^2}{2\sigma^2})$$

等。这些核函数都能够帮助 x 升维，进而达到线性可分的分类效果。由于只需要构建 $K(x, z)$，不需要构建 $\phi(x)$，所以在训练中不给出样本的空间坐标，只给出它们之间的距离，同样能够成功学习出分类规则。这是 SVM 非常强大的地方。

11.8.3 强化学习

与非监督学习和监督学习的工作方式不同，强化学习是一种极为重视反馈的学习过程（如图 11–36 所示）。非监督学习中的聚类，在我们设置了超参数（Hyperparameter）以后，就可以不对学习过程进行干预了（所以称为“非监督学习”）。对于监督学习，我们在一开始就为所有训练样本设置了标签，让算法根据标签进行分类条件的归纳。强化学习是一种基于与环境互动的学习方式。

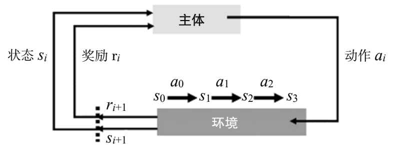


图 11–36 强化学习流程

所谓强化学习是指从环境状态到行为映射的学习，以使系统行为从环境中获得的累积奖励值最大。

在训练主体的过程中，首先赋予主体对外界状态和奖励的感知能力。这里的“奖励”是指广义的奖励，如果是正值就是奖励（Reward），如果是负值就是惩罚（Penalty）。让主体具备感知“利”与“弊”的基本能力是一种很“高级”的学习方式。对周围一切能够被主体感知的因素，包括声音、图像、视频、文字、温度、湿度等，主体每次输出某个动作都会诱发环境产生奖励的反馈。主体通过一系列动作反馈的奖励积累来归纳在某个或某些状态下给予什么样的动

作是最为“有利”或“正确”的。这就是强化学习的本质。

强化学习不是具体的一个算法，而是一类算法的解决方案总括。在这种思想的指导下，从 20 世纪 50 年代开始涌现出了一代又一代强化学习算法体系来解决相应的问题，例如马尔科夫决策过程（Markov Decision Process）、A-Learning、Q-Learning 等著名的算法。

我们通过 Q-Learning 算法的过程来感觉一下强化学习的基本过程吧，看一下伪代码。

```
Initialize Q(S,A) arbitrarily
Repeart (for each episode)
  Initialize S
  Repeat (for each step of episode)
  Choose A from S using policy derived from Q
  Take action A, observe R,S’
    Q(S,A)←Q(S,A)+α[R+γmaxₐ’(S’,A’)-Q(S,A)]
    S→S’
Until S is terminal
```

这段伪代码看上去不太容易理解，下面进行详细的分析。

首先，任意初始化一个 Q 函数的值。重复进行一个训练过程，直到 S 结束。这个 S 是指环境描述向量（State），S 结束就是整个训练过程完毕的一个标识。例如，我们训练一个 2D 飞机游戏的外挂——AI 自动打游戏机器人，S 可以设定为游戏终止状态，例如通关、所有生命都消耗光（如图 11-37 所示）。

然后，就是逐一对 episode 进行训练。这里的“episode”是指一段游戏中的状态。在每个状态中，会设置一个 S（State）向量，在这种游戏中可以是基于游戏元素的向量（例如敌人的种类、坐标、发射的枪弹坐标等），也可以是基于像素的向量（例如直接把当前图像作为向量输入 Q-Learning 算法充当 S 状态）。此时，就可以对主体（飞机）输入动作 A（Action）。动作 A 可以由人输入，也可以由机器通过某种策略输入——当机器学习到一定程度之后，这种方式会大大减少人对学习过程的干预。

在这个过程中，主体会有很多与环境 S 的互动，例如吃下一个加分的物品、发射枪弹消灭敌人或者躲避敌人的子弹等。在这个过程中，我们可以为这些反馈定义其 R 的值，对于一些小的、对通关有帮助的事情，R 值可以为小一些的正数（例如每生还 0.2 秒加 1 分）；而对于那些对通关帮助比较大的事情，我们可以定义为较大的正数（例如消灭一个杀伤力很强的敌人加 50 分）。反之，如果是对通关损害比较小的事情，就为其定义一个绝对值小一些的负数 R（例如未能吃到一个威力比较大的武器减去 20 分）；如果是对通关损害比较大的事情，则为其定义一个绝对值大一些的负数 R（例如中弹死亡减去 1000 分）。每隔 0.2 秒（或其他时间间隔），状态 S 和 Q(S, A) 都要进行更新，更新的内容就是保证 Q 增大方向的策略评价。

在这个过程中通过不断学习和总结，最后 Q(S, A) 会得到一个“在 S 情况下采取某个 A 会有

较大概率获得较大 R”的策略，这个策略就是我们要学习的内容。

如果场景更为复杂，例如 3D 场景，或者有复杂光照、降雨等复杂气候等的场景，S 的定义也会更为复杂。在训练人工智能机器人的过程中，强化训练是使用非常广泛的一种算法体系，也是目前工业界广泛采纳且应用较为成熟的一类方法论（如图 11-38 所示）。

图 11-37　2D 飞机游戏

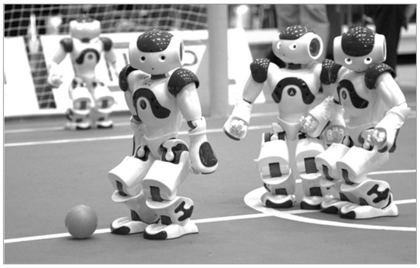


图 11-38　踢足球的机器人

Q-Learning 算法只是一个算法核心，在对外界的感知中很可能需要其他组件或者设备进行辅助，例如使用卷积神经网络作为图像感知的处理途径，以及使用多种复杂的感知系统组合作为 S 输入和 R 输入的处理手段等。整个强化学习的学习思路更像是一个知晓“利”与“弊”、“快乐”与“痛苦”的真实的人，所以基于这种本能刺激得到的策略更像是人在进行主观能动性成分更多的学习。因此，强化学习也被认为是在未来人工智能中最有前景的算法体系。至于其他工作，从理论上讲，就只是计算能力、存储能力和样本数量提升的问题了。

11.9　神经网络——深度学习

神经网络是一种成熟时间相对较晚的算法体系，是机器学习的一个分支领域。其实直到现在，对各种神经网络模型的研究仍在继续，包括其底层实现的数学层面理论依据和推导过程等也在研究过程中。由于神经网络的实验效果比较好，所以在各个深层领域有越来越多成熟的产品落地。

11.9.1　神经元

神经网络的设计思路源于对人脑神经网络的一种模拟尝试。

人的大脑皮层中大约有 140 亿个神经元细胞，而人的全身有数百亿到上千亿个神经元细胞，这是一个非常复杂的神经网络。与计算机不同，神经网络虽然工作频率非常低，但是具有独特的大规模并行分布式结构，因此擅长对抽象事物的学习和模式识别，这些都是神经网络非常吸引人的地方。

如图 11-39 所示，人的神经细胞枝杈很多，一边比较粗大，另一边比较纤细。上端粗大的部分就是细胞体。细胞体上有一些小枝杈，叫作“树突”。细胞体下方像尾巴一样细长的东西叫作“轴突”。不同细胞之间通过树突和轴突传递信息，它们的接触点叫作“突触”。准确地说，一个细胞的轴突通过突触将信号传递给另一个细胞的树突。大量细胞连在一起，形成了复杂的神经网络。

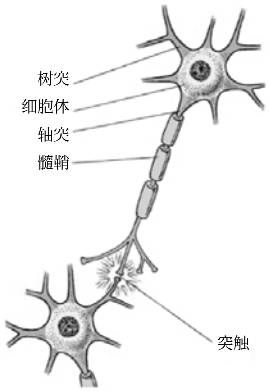


图 11-39 单个神经元细胞

神经细胞通过化学信号进行信息的传递，其传递机制非常复杂。我们使用二进的电子计算机对这一机制进行模拟，就是希望通过一种类似的方式获得像神经细胞那样强大的数据处理效果。

为了模拟神经元细胞，人们设计了一种单个神经元模型（如图 11-40 所示），叫作“Rosenblatt 感知器”，它几乎是最简单的、不可再分的数字神经元。其中，有 n 个维度的输入，每个维度的输入对应一个权重，这样就有 n 个权重，再加上一个偏置 b，最后是一个激活函数。

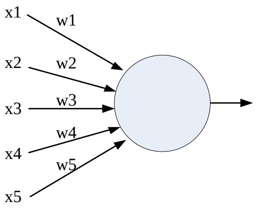


图 11-40 神经元模型

$$u = w^T x$$
$$y = \varphi(u + b)$$

这样，$\varphi(x)$ 分类器函数里的 x 就是一个线性分类器，激活函数则用来对这个输出的“波幅”进行限制，通常限制在 [0, 1]。当然，也会根据不同的需求产生其他激活函数。

在这个表达式中，w^T 是一个 n 维的向量，x 也是一个 n 维的向量，这两个向量做内积再加 b，就形成了一个有关 n 维的 x 向量的线性分类器。如果不约束后面函数值的输出，那么只要给定一个 w^T 和一个 x，就会产生一个实数范围内的数值。一般来说，对于简单场景下的分类模型，可以只尝试使用一个线性分类器来实现，这相当于在一个 n 维空间中寻找一个超平面，并尝试用这个超平面把空间中的向量分成两类。

例如，在一个二维空间中有一个分类器

$$f(\vec{x}) = 2x - y + 2$$ [97]

$$\begin{cases} f(\vec{x}) > 0 & ,1 \\ f(\vec{x}) < 0 & ,0 \end{cases}$$

97 x 和 y 分别是向量 $\vec{x}$ 的两个维度。

相当于在一个二维空间中画一条直线，处于直线两侧的二维向量各属于一个分类（如图 11–41 所示）。

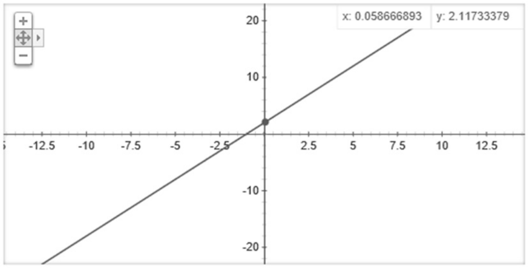


图 11–41　超平面 $2x-y+2=0$

在这样一个分类器模型中，将 x 和 y 带入，就能得到一个 1 或 0 的分类结果。为了让这个投射过程是一个连续可导的函数，通常不会用

$$\begin{cases} f(\vec{x})>0 & ,1 \\ f(\vec{x})<0 & ,0 \end{cases}$$

这种表达式，而是用一个 Sigmoid 函数作为激励函数来完成投射过程（如图 11–43 所示）。

$$z=w\vec{x}+b\text{，}\quad f(z)=\frac{1}{1+e^{-z}}$$

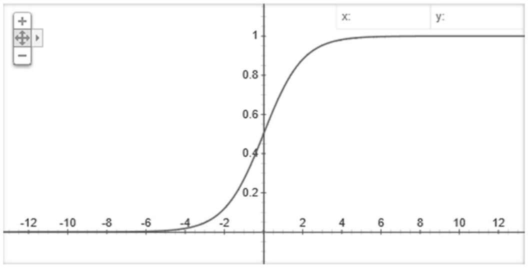


图 11–42　$f(z)=\frac{1}{1+e^{-z}}$

这个图形大家应该都不陌生，在本书中也不是第一次出现了，使用它充当激励函数的好处就是在一个最小的线性分类器单元中用连续可导的函数完成了一个 0, 1 分类问题。

单个神经元的工作方式与逻辑回归几乎没有区别，训练过程也一样。这只是最简单的神经元设计方式。神经元的设计方式其实不止这一种（例如，在基于核方法的径向基函数网络中使用的就不是这种方式），激励函数也远不止 Sigmoid 函数这一种。

11.9.2 BP 神经网络

BP 神经网络（Back Propagation）是所有神经网络中最为传统的一种，也是在数学模型上相对容易理解的一种。

2 层全连接网络（如图 11-43 所示）有一个输入层用来接收输入的完整向量，有多少个维度就有多少个输入项。隐含层可以是 1 层，也可以是多层全连接的状态，每层上每个节点的输入都来源于上一层所有神经元的输出。在这个矩阵里，隐含层和输出层是处理数据的关键。

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & \cdots & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

这样，一个 m 行 n 列的 BP 神经网络可以简记作一个 m 行 n 列的矩阵，这也是训练后得到的结果。

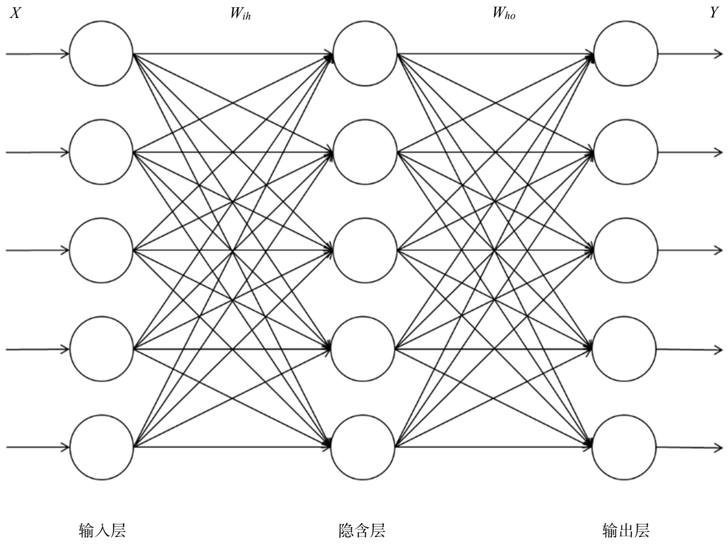


图 11-43 2 层全连接神经网络

11.9.3 损失函数

和一般机器学习中的 SVM、线性回归一样，神经网络中的损失函数也是用每个样本 x_i 带入后的输出值 $f(x_i)$ 及样本中的 y_i 的差异的累加值来描述的。这种差异的描述方式不止一种，常见的方式列举如下。

绝对值损失函数：

$$C=\sum_{i=1}^{n}\left|f(x_i)-y_i\right|$$

平方损失函数：

$$C=\frac{1}{2}\sum_{i=1}^{n}(f(x_i)-y_i)^2$$

交叉熵损失函数：

$$C=-\frac{1}{n}\sum_{i=1}^{n}[y_i\log(f(x_i))+(1-y_i)\log(1-f(x_i))]$$

在寻求损失函数极小化的过程中，目前工业领域应用效果比较好的方法就是通过梯度下降或者随机梯度下降的方式来寻找极小值。对每个 w_{ij} 求偏导，会得到如下矩阵。

$$\frac{\partial C}{\partial w}=\begin{bmatrix}\frac{\partial C}{\partial w_{11}} & \frac{\partial C}{\partial w_{12}} & \cdots & \frac{\partial C}{\partial w_{1j}}\\ \frac{\partial C}{\partial w_{21}} & \cdots & \cdots & \frac{\partial C}{\partial w_{2j}}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{\partial C}{\partial w_{i1}} & \frac{\partial C}{\partial w_{i2}} & \cdots & \frac{\partial C}{\partial w_{ij}}\end{bmatrix}$$

每次通过更新 w_{ij} 进行迭代，经过多次迭代找到极小值的 w_{ij}。

$$w_{ij}=w_{ij}-\eta\frac{\partial C}{\partial w_{ij}}$$

$$b_{ij}=b_{ij}-\eta\frac{\partial C}{\partial b_{ij}}$$

其中，η 是学习率，即更新时移动的步长基数；$\frac{\partial C}{\partial w_{ij}}$ 和 $\frac{\partial C}{\partial b_{ij}}$ 分别对矩阵中的第 i 行第 j 列的 w 和 b 求偏导数。这种更新方式会使 *Loss* 函数优化过程在最初斜率较大的部分移动更新幅度较大，而在临近极小值的部分自动将更新幅度调小。至于每个 w_{ij} 究竟应该移动多少，我们通过推导相对容易的平方损失函数 $C=\frac{1}{2}\sum_{i=1}^{n}(f(x_i)-y_i)^2$ 来看一看。这里的“*C*”代表“Cost”，与“Loss”同义。根据链式法则，有

$$\frac{df(g(x))}{dx}\Leftrightarrow g=g(x),\frac{df(g(x))}{dx}=\frac{df(g)}{dg}\cdot\frac{dg(x)}{dx}$$

则对最后一层上任意一个神经元的 w_0，都有

$$\frac{\partial Loss(w_0,b_0)}{\partial w_0}=\sum_{i=1}^{n}(f(w_0,b_0,x_i)-y_i)\cdot\frac{\partial\sum_{i=1}^{n}f(w_0,b_0,x_i)}{\partial\sigma}\cdot\frac{\partial\sigma}{\partial w_0}$$

别忘了，这里还有一个激励函数 $\sigma=\frac{1}{1+e^{-(w_0x+b_0)}}$。

同理，对最后一层上任意一个神经元的 b，都有

$$\frac{\partial Loss(w_0,b_0)}{\partial b_0}=\sum_{i=1}^{n}(f(w_0,b_0,x_i)-y_i)\cdot\frac{\partial\sum_{i=1}^{n}f(w_0,b_0,x_i)}{\partial\sigma}\cdot\frac{\partial\sigma}{\partial b_0}$$

对最后一层来说，它的前一层的任意一个神经元的输出都是其函数的输入，那么对于其前一层的任意一个神经元的 w_1 和 b_1 来说，仍然可以用链式法则来推导。

通常这种损失函数得到的值非常大（可能是几百万，上亿也毫不奇怪），尤其是在几百万维的图片识别训练中——这与它的定义有关。在它逐步减小的过程中，通过判断损失函数值超过某个值来决定终止通常是很不明智的，原因有如下两个。

第一，在这个收敛过程中，损失值的大小没有具体的物理解释，所以无法通过指定一个“较小”的损失值来确保召回率和准确率。

第二，如果通过不断的收敛，最终使损失函数的值减小为 0，通常也会产生严重的过拟合现象，这种现象在模式识别中会显得格外严重。

所以，在训练过程中，通常会把整个训练集按一定的比例分配，一部分作为训练集，另一部分作为验证集。例如，按 4∶1 的比例分配，其中 80% 的样本用来做训练，20% 的样本用来做验证（抑或其他比例，70% 的样本用来做训练，30% 的样本用来做验证）。训练过程是：在用 80% 的样本做训练的同时，不断用当前迭代产生的 w_{ij} 矩阵去对 20% 的样本做分类，直到分类结果中召回率和准确率没有明显的提高为止，否则就会看到一个现象——随着训练时间的延长（Epoch 增多），验证集中的损失函数值反而会增加（召回率和准确率会下降）。此外需要注意，样本的数量要足够多（因为样本数量太少本身就非常容易造成过拟合）。

11.9.4 非线性分类

神经网络最强大的地方在于它对复杂的线性不可分问题的处理能力，这是 SVM 无法企及的，也是神经网络最大的优势。在深度神经网络（深度学习）领域，通过对网络进行合理的设计，通常能够对音乐、图像、视频等模式识别领域的分类有比较好的召回率和准确率。

为什么深度神经网络有这么好的非线性分类能力呢？我们来看看样本通过网络的时候究竟发生了什么。

如图 11-44 所示，在一个二维空间中有一个六角形，如果要用普通的线性 0, 1 分类器对这

样一个“不规则”图形做分类，是无论如何都找不到一条（一个）可以把空间恰好分为两部分的直线（超平面）的——一部分是组成六角形的部分，另一部分不是组成六角形的部分。这种情况下的 BP 网络应该怎么设计呢？

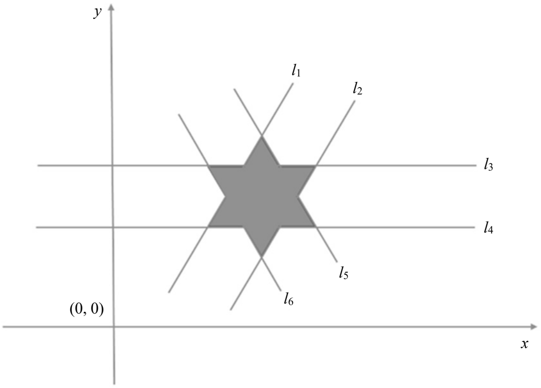


图 11–44　非线性分类

我们可以考虑设计一个 3 层的神经网络（如图 11–45 所示），输入层只有 2 个神经元，分别接收 x 和 y 的坐标值。隐含层有 2 层，在物理上可以解释为拟合出 l_1 到 l_6 各自的表达式，以及相互之间的“与”或“非”关系。输出层有 1 层，用来最终输出 0（否）或 1（是）。

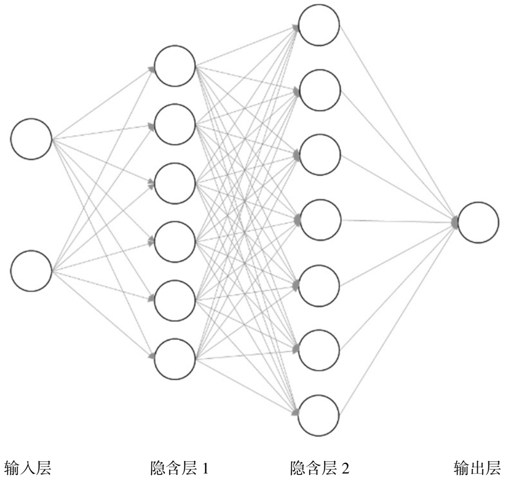


图 11–45　3 层 BP 网络

这样一个六角形区域中的点应该怎样表示呢？

设 6 个超平面的表达式分别为 $l_1(x, y)$、$l_2(x, y)$、$l_3(x, y)$、$l_4(x, y)$、$l_5(x, y)$和 $l_6(x, y)$，则这个六角形可以转化成中心的 1 个正六边形和旁边的 6 个正三角形的表达式的并集，即

$$
\begin{aligned}
&F(x,y)= \\
&(l_1(x,y)<0 \cap l_2(x,y)>0 \cap l_3(x,y)<0 \cap l_4(x,y)>0 \cap l_5(x,y)<0 \cap l_6(x,y)>0) \\
&\cup \\
&(l_1(x,y)<0 \cap l_5(x,y)<0 \cap l_3(x,y)>0) \\
&\cup \\
&(l_2(x,y)<0 \cap l_3(x,y)<0 \cap l_5(x,y)>0) \\
&\cup \\
&(l_2(x,y)<0 \cap l_5(x,y)<0 \cap l_4(x,y)>0) \\
&\cup \\
&(l_4(x,y)<0 \cap l_2(x,y)>0 \cap l_6(x,y)>0) \\
&\cup \\
&(l_1(x,y)<0 \cap l_6(x,y)<0 \cap l_4(x,y)>0) \\
&\cup \\
&(l_6(x,y)>0 \cap l_1(x,y)>0 \cap l_3(x,y)<0)
\end{aligned}
$$

能够表达这样一种关系的神经网络是很容易设计的。

从理论上讲，这个网络中所拥有的 VC 维已经足够把刚刚的非线性分类问题转换成线性分类和“与”或“非”判断的问题组合了。单个神经元的工作方式是线性分类器的工作方式，这个观点刚刚已经讨论过了，那么“与”或“非”计算是怎么实现的呢？

对于一个有 2 个维度输入的神经元来说，“与”运算的构造要达到如表 11–8 所示的效果。

表 11–8　与运算

维度 1 输入	维度 2 输入	输　出
1	1	1
1	0	0
0	1	0
0	0	0

那么，分别构造一个 w 和一个 b，使其满足上面这个约束，其实也很容易。取 $w=[20,20]$，$b=-30$（取法不止 1 种）。

带入 $f(x)=wx+b$，会得到如表 11–9 所示的结果。

表 11–9　$f(x)=wx+b$ 输入与输出（1）

维度 1 输入	维度 2 输入	输　出
1	1	10
1	0	–10
0	1	–10
0	0	–30

对这样的结果，通过 Sigmoid 激励函数 $f(z)=\dfrac{1}{1+e^{-z}}$ 很容易得到与期望一致的 1 和 0 的输出结果。

同样，构造一个“或”关系计算也很容易，如表 11–10 所示。

表 11–10　或运算

维度 1 输入	维度 2 输入	输　　出
1	1	1
1	0	1
0	1	1
0	0	0

那么，分别构造一个 w 和一个 b，使其满足上面这个约束，其实也很容易。取 $w=[20,20]$，$b=-10$（取法同样不止 1 种）。

带入 $f(x)=wx+b$，会得到如表 11–11 所示的结果。

表 11–11　$f(x)=wx+b$ 输入与输出（2）

维度 1 输入	维度 2 输入	输　　出
1	1	30
1	0	10
0	1	10
0	0	–10

对这样的结果，再通过 Sigmoid 激励函数 $f(z)=\dfrac{1}{1+e^{-z}}$，也很容易得到与期望一致的 1 和 0 的输出结果。

同样，构造一个“非”运算也很容易，如表 11–12 所示。

表 11–12　非运算

维度 1 输入	输　　出
1	0
0	1

那么，分别构造一个 w 和一个 b，使其满足上面这个约束。取 $w=[-20]$，$b=10$（取法同样不止 1 种）。

带入 $f(x)=wx+b$，会得到如表 11–13 所示的结果。

表 11–13 $f(x)=wx+b$ 输入与输出（3）

维度 1 输入	输　　出
1	–10
0	10

对这样一个结果，再通过 Sigmoid 激励函数 $f(z)=\frac{1}{1+e^{-z}}$，仍旧可以得到与期望一致的 1 和 0 的输出结果。

当从多个维度输入的时候，通过 w 矩阵中 0 值的设置可以“忽略”一些对输出没有影响的维度。通过多个隐含层的叠加，可以构造数量庞大的线性分类器和“与”或“非”计算的组合，进而实现非线性分类器。

所以，在刚刚讨论的 3 层 BP 网络中，第 1 层的 6 个节点可以写成一个线性分类器的表达式，第 2 层的 7 个节点分别是

$$l_1(x,y)<0\cap l_2(x,y)>0\cap l_3(x,y)<0\cap l_4(x,y)>0\cap l_5(x,y)<0\cap l_6(x,y)>0$$

$$l_1(x,y)<0\cap l_5(x,y)<0\cap l_3(x,y)>0$$

$$l_2(x,y)<0\cap l_3(x,y)<0\cap l_5(x,y)>0$$

$$l_2(x,y)<0\cap l_5(x,y)<0\cap l_4(x,y)>0$$

$$l_4(x,y)<0\cap l_2(x,y)>0\cap l_6(x,y)>0$$

$$l_1(x,y)<0\cap l_6(x,y)<0\cap l_4(x,y)>0$$

$$l_6(x,y)>0\cap l_1(x,y)>0\cap l_3(x,y)<0$$

的判断分类线性模型，最后一层的节点是上述 7 个条件的“或”条件判断线性分类器。

在训练的过程中，只要对各个 w 和 b 的初始化合理，使用梯度下降法是比较容易找到极小值的。

11.9.5 激励函数

除了前面提到的 Sigmoid 激励函数以外，还有其他一些激励函数可供使用。

激励函数在选择的时候有一些先决条件，第一是可导性，第二是单调性（因为连续可导性和单调性可以保证在使用梯度下降法迭代的过程中逐步找到极小值）。

还有一种与 Sigmoid 函数的图像相近，曲线也呈 S 形的激活函数，叫作 tanh 函数（如图 11–46 所示）。

$$\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$$

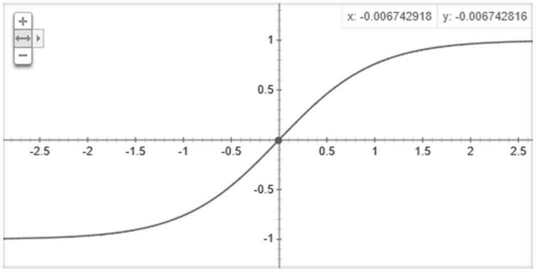


图 11–46　tanh 函数图像

从图像上看，tanh 函数的取值范围是 (–1, 1)。从限制输出波幅的特性上来说，它与 Sigmoid 函数都能起到比较好的作用。然而，这类 S 形激活函数在利用梯度下降法进行学习的时候，在学习率 η 确定的情况下，会产生比较严重的"梯度消失"（在曲线两端近乎直线的部分由于斜率太低而导致 w 的更新速度极慢的现象），所以现在这两类激励函数在工业上的应用没有之前那么广泛了，取而代之的是在迭代学习中收敛速度较快的激励函数，例如 Softplus 和 ReLu。

Softplus 函数的表达式如下。

$$y = \log(1+e^x)$$

Softplus 函数在第一象限的斜率很稳定（如图 11–47 所示），所以它在第一象限会收敛得很快，也很稳定。

传统的 ReLu 函数（Rectified Linear Units）的表达式如下。

$$y = \max(x, 0)$$

如图 11–48 所示是函数 $y = x$ 在第一象限的部分及函数 $y = 0$（$x < 0$）的部分。这个函数在第一象限的收敛特性非常好，但是到了 $x < 0$ 的部分就没有任何学习能力了。因为导数是 0，所以要使用这种激活函数的改进版 Leaky ReLu 对 $x < 0$ 部分的学习能力进行改进。

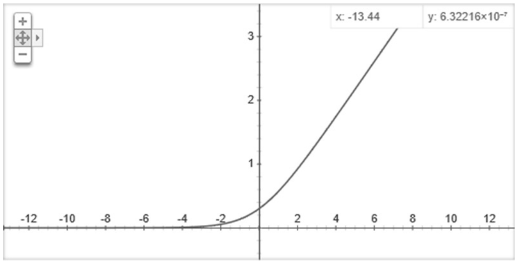


图 11–47　Softplus 函数图像

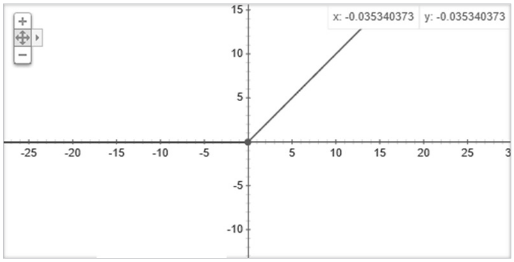


图 11–48　ReLu 函数图像

Leaky ReLu 函数（如图 11–49 所示）的表达式如下。

$$f(x)=\begin{cases} x\ ,x\geq 0 \\ \alpha x,x<0 \end{cases}$$

其中，α 是一个取值较小的实数，例如 0.2。

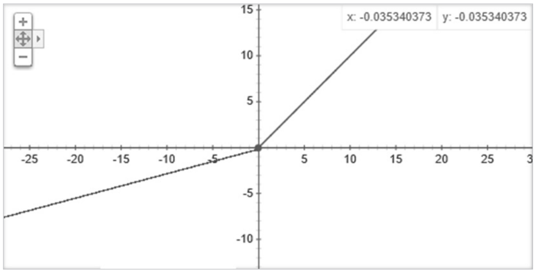


图 11–49 Leaky ReLu 函数图像

除了上述这些激励函数以外，在其他文献中还提到了 ArcTan、Bent Identity、Gaussian 等激励函数及其改进版本。我们很难说清这些激励函数孰优孰劣。由于函数本身的特性导致了在不同场景中不同的学习效果，所以目前在各种成型的深度学习网络中，ReLu 函数的使用率还是很高的。

11.9.6 卷积神经网络

在神经网络这种算法设计方式中，有一种叫作“卷积神经网络”（Convolutional Neural Network，CNN）的子类别。

从工作流程来看，卷积神经网络与前面提到的 BP 神经网络类似，向量输入都是从前向后传递，都是通过训练集来获得损失函数的定义，并通过求偏导数来调整每个神经元的权值矩阵 w 和 b 从而获得损失函数更小的解，在流程上也没有太大区别。但是，卷积神经网络中多了一个组件——卷积层。卷积神经网络与传统的 BP 神经网络相比，在模式识别方面的处理能力上优势更大，训练收敛速度更快，准确率更高。

卷积层是由卷积核组成的一种以采样为目的的处理层。

以图像处理为例，一幅包含 $m\times n$ 个像素的图片在处理时实际上被当成了一个 $m\times n\times 3$ 维度的向量（“3”是指 RGB 通道，每个通道有 256 种取值）。如果要对这么大的一个向量设计全连接的 BP 神经网络，需要训练的 w 矩阵中的权值数量是非常惊人的。而卷积神经网络通过自己独特的处理技巧——权值共享，可以有效降低计算量。

我们以 VGG–16[98] 为例讲解，如图 11–50 所示。

98 VGG 的卷积神经网络模型除此之外还有 VGG–19 等。

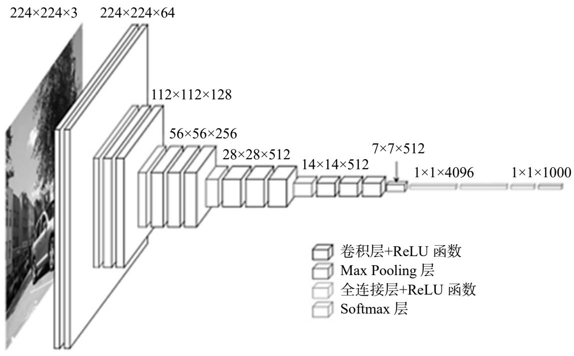


图 11-50　VGG-16 Model

“VGG”是“Visual Geometry Group”的缩写。Visual Geometry Group 是隶属于英国牛津大学工程科学系的一个组织。“16”是指其中有 16 个带有参数的网络层。

“Convolution”就是卷积层，使用卷积核进行卷积操作（如图 11-51 所示）。卷积核的设计可以说是不一而足的，由于其目的是采样，所以采样的逻辑可以根据场景的需要进行调整。在这个例子中，我们看到的是将一个连续 3×3 范围内的点值做加和。这相当于 $f(x)=wx+b$，每个点组成的向量是 x，w 是采样时赋予的系数。在这之后，每个卷积核会通过一次 ReLu 函数完成激励，也就是说，每个 3×3 的 9 维向量都需要做一次这样的映射：

$$f(x)=\log(1+e^{wx+b})$$

如图 11-51 所示，右侧那幅由左侧图片经过卷积核从左到右、从上到下不断扫描而产生的特征图叫作“Feature Map”，用来保存左侧卷积核扫描的结果。Feature Map 中的数据相当于通过某种卷积核采样后采集到的“特征值”，而这种特征值存在一个问题——很难有一个确切的物理解释。

在 VGG-16 这样的卷积神经网络中还有一种叫作“Max Pooling”的层（如图 11-52 所示），这种层不一定会出现在每个卷积神经网络模型中。它的作用是对临近的几个点进行取最大值的操作，实际上是在进行一种有损压缩。有些卷积神经网络中会设计其他的 Pooling 层，例如不是取最大值，而是取平均值（Average Pooling）等。学术界对这些有损压缩方式的优劣到现在为止也是莫衷一是，通常只能在具体的场景中去比较不同的选择产生的不同的召回率和准确率，然后得到一个特定场景中的结论。不少较新的卷积神经网络中根本没有 Pooling 层。

Full Connected 层也叫“全连接层”。这一层上的处理方式和前面提到的 BP 神经网络中的连接方式是一样的，第 2 层的每一个节点都与上面一层的每一个节点相连接。这种连接方式的含义和 BP 神经网络中的含义一样，每个上一层的计算结果都会对下一层中每个节点的输出起作用。而且，在这一层上同样是用 ReLu 的方式

$$f(x)=\log(1+e^{wx+b})$$

对每个节点的输出值做激励。

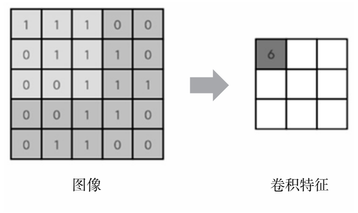


图 11-51 卷积操作

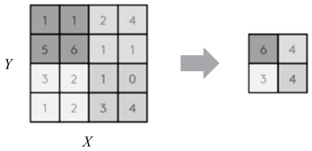


图 11-52 Max Pooling 层

Softmax 层使用的是一种回归模型（如图 11-53 所示）。

$$\sigma_i(z)=\frac{e^{z_i}}{\sum_{j=1}^{m}e^{z_j}},\quad i=1,2,\ldots,m$$

$$z_i=w_ix+b_i$$

x 是全连接层输出的完整向量。

最后输出的一层表示预测分类，其中的每一个节点都是某个预测分类（从 1 到 j）概率的输出值。

z_i 是一个向量，也就是最后一层全连接层的输出向量 x 与第 i 个分类节点的 w_i 的内积与偏置 b_i 的加和。$\sigma_i(z)$ 表示向量 x 属于 i 分类的概率。

通过计也可以得到 $\sum_{i=1}^{j}\sigma_i(z)=1$，含义为一个向量在各个分类中的概率加和为 1。

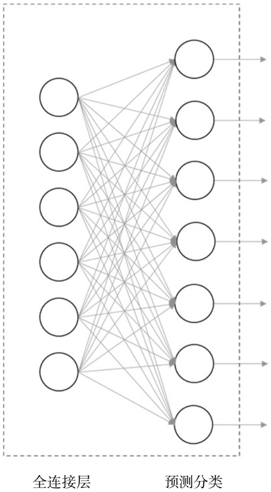


图 11-53 Softmax 层

VGG-16 只是众多使用卷积层的网络模型之一，还有很多优秀的网络模型中会用到卷积层及与其他神经元的混编模型。卷积层的目的是做某种特征的提取（我们只要理解这一点就可以了）——帮助人们自动进行特征提取，并通过训练找到更合理的提取方式。训练的过程则是通过样本和输出值描述的损失函数，最终找到每一个卷积层上的 w 和 b，以及最后 3 个全连接层每个节点的 w 和 b。

11.9.7 循环神经网络

下面介绍一种循环神经网络（Recurrent Neural Networks，RNN）。RNN 与其他网络的不同之处在于，其中的“循环”部分能使网络具有“记忆”功能（如图 11-54 和图 11-55 所示）。

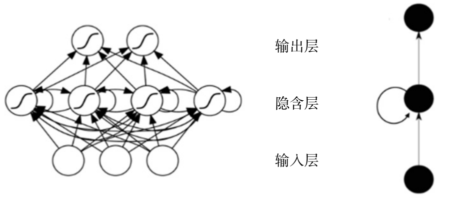


图 11–54　RNN 网络结构（1）

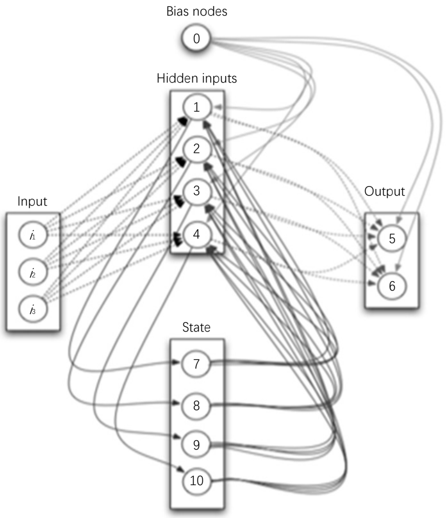


图 11–55　RNN 网络结构（2）

在观察一个向量序列 x_i 的过程中，x_t 会通过这个这个网络产生一个输出，而这个输出会在 x_{t+1} 进入网络的同时作为一个输入项和当前的 x_{t+1} 一起输入到网络中。从时间上看，如图 11–56 所示。

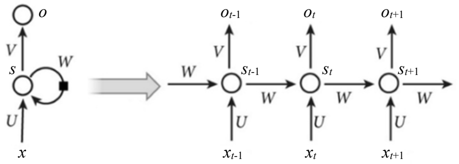


图 11–56　RNN 网络结构（3）

从 RNN 网络的设计中不难看出，它的思路是把从 x_1 到 x_t 的所有影响都叠加到 x_{t+1} 的训练时刻，应该说还是很巧妙的。所以，这类设计非常适用于上下文相关的序列预测场景，例如自然语言处理、股票行情分析等领域。标准的隐马尔可夫模型也可以用来做序列前后转换概率统计与预测。但隐马尔可夫模型只认为当前状态和前一个状态有关，而不会关注再向前延伸的其他状态，这是两者最大的不同。

在 RNN 模型中会遇到一个很严重的问题，就是梯度消失。梯度消失问题当然不是 RNN 的专利，不过 RNN 会在时间这个梯度上发生梯度消失问题，这样序列中离现在较远的状态对当前输出的刺激实际上是极为不足的。为了克服这个问题，改进 RNN 的学习性能，产生了 LSTM 衍生算法（如图 11–57 所示）。

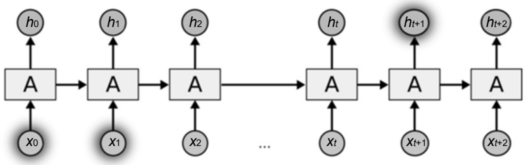


图 11–57 LSTM 序列

基于 LSTM（Long-Short Term Memory）算法实现的神经网络也叫作“长短期记忆人工神经网络”，它是 RNN 网络的一个重要变种（如图 11–58 所示）。LSTM 算法训练的对象就是 x_i 和 h_i 的序列对，x_i 是输入向量，h_i 是输出向量。

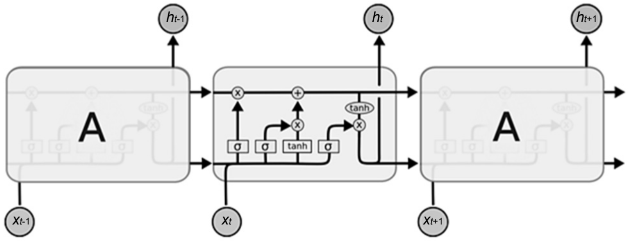


图 11–58 LSTM 网络结构

在一个 LSTM 网络中有这样几个结构，分别如图 11–59、图 11–60、图 11–61 和图 11–62 所示。σ 就是 Sigmoid 函数，有

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f)$$

f_t 的激励值来源于 h_{t-1} 和 x_t 两个元素，也就是当前的 x_t 向量和前一个输出值 h_{t-1}。这个输出的 f_t 取值范围为 (0, 1)，输出到上面一条横线后通过一个乘法器与上一次的输出值 h_{t-1} 相乘。这个部分叫作“忘记门”。所谓“忘记”就是指 h_{t-1} 输入的值在乘以一个取值为 0 到 1 之间的百分比之后还剩下多少。如果是 0，那就是完全忘记；如果是 1，那就是完全记住。

i_t 这个值的产生方式与前面的 f_t 是一样的，为了区别，下标用 t 来表示。

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

$\tilde{C}_t$ 的含义是通过 h_{t-1} 和 x_t 向量的刺激产生一个取值范围为 (–1, 1) 的值。

$$\tilde{C}_t = \tanh(w_c[h_{t-1}, x_t] + b_c)$$

i_t 和 $\tilde{C}_t$ 的乘积生成一个在 (–1, 1) 之间的值作为权重。

传递到下一个状态的 C_t 是由 $\tilde{C}_t$ 和 C_{t-1} 线性叠加得到的。这个部分是 LSTM 和传统的 RNN 区别最大的地方，C_{t-1} 及以前的状态可以通过这样一个普通的线性叠加方式传递到当前状态。

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

最后，这个结构决定输出 o_t 和 h_t。

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

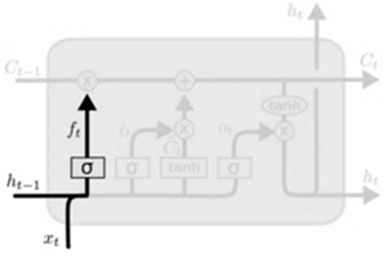


图 11-59　LSTM 结构（1）

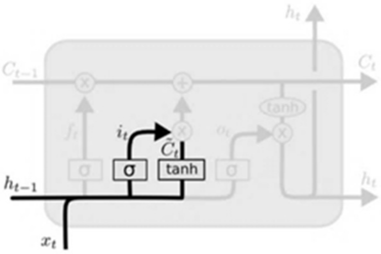


图 11-60　LSTM 结构（2）

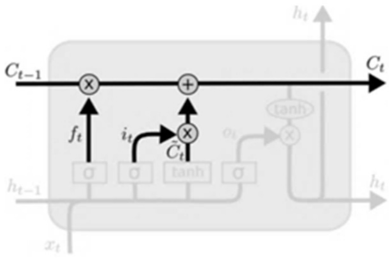


图 11-61　LSTM 结构（3）

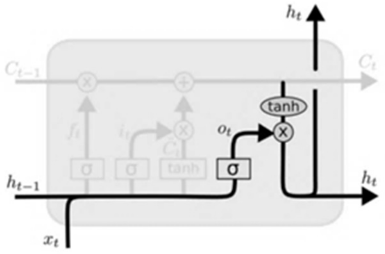


图 11-62　LSTM 结构（4）

在实际应用中，通常根据输入的维度来决定 LSTM 单元的使用数量。例如，在 NLP 领域通常滑动窗口会比较大，要至少能够覆盖整个句子。所以，这种情况下可能会使用上百个 LSTM 单元一起工作。

11.9.8　小结

目前，国内和国际每年发表的关于神经网络的研究论文数以千计，在神经元设计、激励函数选择、连接方式设计、网络组成等方面，很多研究人员都在不断尝试。由于深度学习网络中蕴涵的 VC 维极高，所以它适用的非线性可分的场景依然很多，在实验层面的结果也比较令人满意。

不过，我们从训练方式中不难看出，要想让深度学习网络学习出来的模型健壮，就必须给它非常多的训练样本，以免产生过拟合现象，而过拟合现象几乎一直在发生。深度学习网络几乎是在以穷举的方式学习样本中蕴涵的维度信息，即使可以通过加大样本量的方式来使网络对没有泛化能力的维度特征尽可能不敏感，但其他样本给模型带来的泛化能力高低仍旧取决于

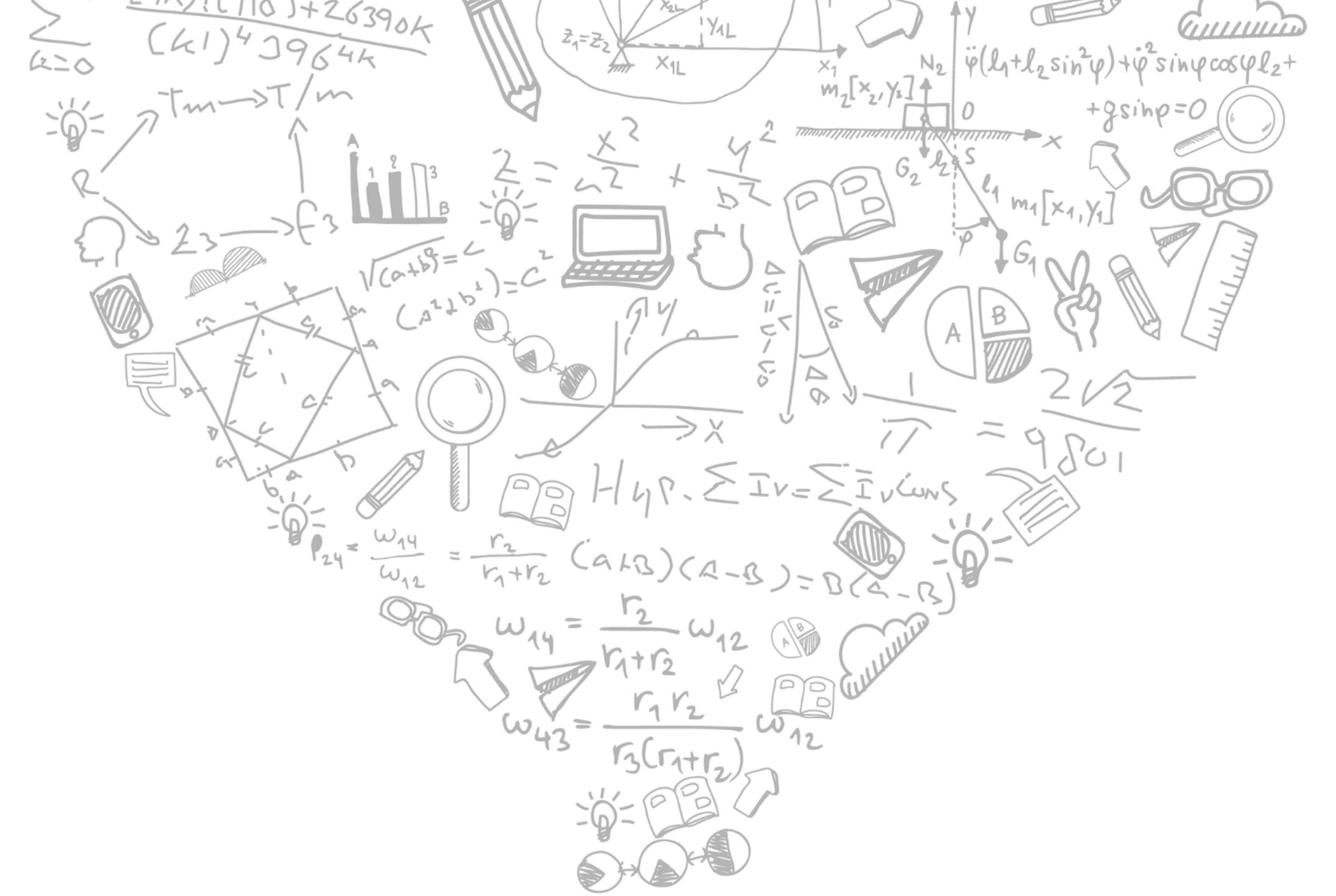
样本数量的多少。这也是深度学习目前仍无法胜任很多应用领域工作的重要原因，我们必须对其有足够的重视。

11.10 本章小结

算法学作为一门使用计算机资源调度技术来处理和解决问题的学科，要求研究人员除了在数学层面要有足够高的知识水平，还要充分掌握和理解计算机的离散特性和资源协调特性。

在算法学里，从微观到宏观，贯穿着处理问题的哲学思想。变通处理，化无穷为有穷，化复杂为简单，化有中心为无中心，时空互化，统筹有限资源——对这些思想的贯彻，尤其是在更为宏观层面的理解与运用，往往会有事半功倍的效果。

在学习过程中，要注意算法本身就是一种思想或者策略，当它们应用于不同场景时会有很多变种出现。这些变种在思想层面继承了经典算法的衣钵，只是在大量处理细节上有很多偏重和技巧，从而形成了算法族或算法系列。面对纷繁变化的算法，不要过早陷入细枝末节，还是应该本着从干到枝、从枝到叶的逐步深入的原则来学习。算法在整个数据科学体系里也是最能体现“厚积薄发”的一个分支学科，在研究说过程中一定要注意积累、归纳和总结。



实践篇

在一个完整的数据生命周期中，对工程实践应用进行归纳性的讨论，看看在数据收集、数据存储、数据统计、数据建模、数据可视化和数据决策中各种方法的优劣和取舍的原则。这个过程说起来非常有趣，有很多的方法可以尝试，同时又有一定的定势可循。

第 12 章　数据采集

12.1　数据的源头

数据采集是整个大数据实践的第一步。在这里我特意用了“大数据”这个词，并不是因为这个词时髦，而是希望用这样一个概念笼统的词汇来指代一切沿着数据的生命周期进行生产实践的行为。

现在绝大多数的大数据应用场景都是在计算机上实现的。在个人 PC 上通常使用 Windows 操作系统，这种操作系统相信大家一定不会陌生。经过 20 多年的进化，Windows 上已经有了非常多的伴生软件，例如处理数据的、处理图像的、处理文本的、处理视频的，非常丰富。

在个人 PC 上我们可以进行大数据实践吗？可以。例如，使用 Excel 就可以对少量（几万条规模的数据）的数据进行统计并生成图表，非常方便。数据的载体是 Excel 文件，或者是 .csv 格式的方阵式文本文件。Excel 虽然现在非常普及，但它并不是行业标准，更不是专业的数据处理软件。Excel 功能的发展，以及其他软件对 Excel 文件格式的兼容，在我看来更多的是一种不得已，是一种由数据承载惯性带来的结果。

对拥有少量数据的用户，尤其是那些对实时性及数据量没有要求的用户来说，这是一种性价比极高的方式。当然，这不是企业应用的普遍场景，更多的企业在数据收集的过程中采用了自动或者半自动传输的方式。

数据的产生是无处不在的，对世界上任何一个事物在任何一瞬间的行为描述，只要你愿意，都可以量化成一个向量。所以，数据的产生从其最源头来说就像“永动机”一样一刻不停，持之以恒。不过，人类能够收集且有意愿收集的数据相比之下就非常少了，而且“本着解决实际问题的目的去收集数据”这一行为本身就会优先收集潜在价值较高的数据，即使是在对利益追逐不明显的学术研究领域也有同样的特性。

数据的收集，无论是哪种形式，一定是类似这样的流程：“感知”→“传输”→“保存”（如图 12-1 所示）。

图 12–1　数据的收集流程

在一些单一、孤立、数据量小或同一情况不会重复出现的场景中，通常直接由人拿着量具进行观测就可以了。除了前文提到的测量长度、温度等的量具以外，也有用于计数的量具。如图 12–2 所示，这种机械式手持计数器读者可能在一些对运营比较重视的场所见到过，或者在飞机上看到空中小姐用它来清点乘客数量。这样的测量行为虽然经常反复出现，但都是非常典型的单一、孤立、数据量小的场景，所以人工测量方式尚且适用。测量完成后，就可以把读数记录下来，誊写到 Excel 或者文本文件中以备后续统计工作之用。这是一个由人手动完成的“感知”、“传输”、“保存”一体化的过程。

图 12–2　机械式手持计数器

在数据量大、数据收集点较多的情况下，再由人来完成这样的采集工作成本就非常高了，而且容易出错。所以，现在的大数据应用场景中都是尽可能使用各种程序来完成工作。

12.2　日志的收集

日志是指事件记录——不论这种行为是一种更贴近人类生活的行为记录（例如消费行为记录、参观行为记录），抑或更贴近计算机底层操作的行为（例如攻击告警记录、磁盘 I/O 记录）。本书提到的日志都是广义的日志，而非服务程序在服务器磁盘上留下的日志文件。

日志发生的位置通常是分散的。例如，在一个连锁便利店的销售系统中，所有的销售记录和盘点记录都分散存放在各个营业点，一般来说会在白天营业结束后通过一些离线的操作进程归集到数据中心来存档。这就是典型的数据延时收集。

从分散的终端集中收集数据的解决方案不止 1 种，让我们来看看都有哪些方式。

对终端不能或不便存储日志的场景，通常在事件发生后立刻将日志数据上报，产生的是单条数据日志。我们也称这种方式为“日志流”（Log Streaming）。

对终端有一定存储能力的场景，通常在其本地保存一份日志以备查阅，同时通过其他方式将日志上传。因实时性不同，分为实时上传和延时上传两种。实时上传的情况和单条数据日志上报的场景类似，而延时上传的情况和前面说的存档销售记录和盘点记录的方式类似。

12.2.1　实时上传

1. 上传方式与问题

实时上传的常见方式有如下 3 种。

方式 1：在终端使用 DTU 上传

DTU（Data Transfer Unit）是一种数据传输模块，以无线 DTU 居多（如图 12-3 所示）。

图 12-3 无线 DTU 模块

DTU 通常可以当成一个“黑匣子”来使用，有“入口”和“出口”两个数据接口。“入口”可以连接 RJ-45、COM、1394、USB、同轴电缆等多种数据接口，通常用来接收由电子探测装置传送过来的数据。常见的电子探测装置包括温度、压力、气体浓度、磁场强度等的探测设备。“出口”一般有 CDMA、GPRS、IEEE 802.11x（Wi-Fi）等多种协议的接口，通过 IP 协议栈把数据送入互联网。这种设备在矿山、自然环境监测、民用远程抄表等场景中经常使用。

方式 2：在终端使用 Wi-Fi 或有线以太网上传

使用 Wi-Fi 上传的场景在我们身边就更常见了。

每个人使用的智能手机和 PC 上都安装了许多软件，每个软件只要拥有本地网络接口的使用权限，就可以进行数据的上传和下载。这些软件中的大多数每天都会与其后台所连接的服务器通信，报送一些从本机上扫描到的信息，包括且不限于本地的 MAC 地址、IP 地址、CPU 资源、内存资源、安装软件列表，甚至一些信息的输入记录等。

方式 3：在服务器端使用以太网上传

在使用互联网对外提供服务的企业中，会有相当数量的服务器产生大量的日志。这里包括这些服务器的访问日志，也包括其自身系统的工作日志等。由于服务器本身具有一定的日志存储能力，所以自然允许实时和延时两种形态的日志传送。对于需要实时传送的服务器来说，可以考虑使用一些开源工具帮助收集和发送日志。

在每一台希望享受日志告警分析服务的主机上部署一套第三方程序是需要勇气的，这里有太多的未知风险，谁也不知道自己是不是下一个特洛伊木马[99]的受害者。因此，更为实际的方式是在自己授权的情况下主动开放不涉密的日志内容给服务提供方。这么说可能太拗口，其实就是在服务器上安装一个软件，由这个软件收集指定的日志内容，并实时把日志内容发送到提供日志告警分析服务的相应监听端口供其收集。只要本地的收集进程是可信的，不会随便收集指定日志内容以外的任何信息，那就非常安全。

所以，在这个环节很多公司（尤其是互联网公司）都喜欢使用一些开源的工具，对下载的源代码进行一些个性化的修改和验证，确认没有问题后编译使用。这种方式也被认为是目前性

99 木马软件（Trojan Horse Program），含有恶意代码的程序。

价比最高的数据采集方式。这类软件的选择范围也比较广，业界常用的有 Fluentd[100]、Flume[101]、Heka[102] 等。

以 Fluentd（如图 12-4 所示）为例，这种工具可以安装在任意的主机上，对系统日志、Web 服务日志等各种日志进行监控，只要发现新的日志内容就会立刻收集起来。在收集的过程中，这些日志可以经过第一次处理，也就是在没有发出去之前这些先过滤日志，根据过滤条件来选择对新发现的一条日志是做丢弃处理、发往其他的服务器地址，还是只在本地其他文件上转存等，十分方便。对于互联网公司，尤其是初创的互联网公司来说，这种开源工具能够节省大量的开发成本，让开发人员把精力向业务方向集中。

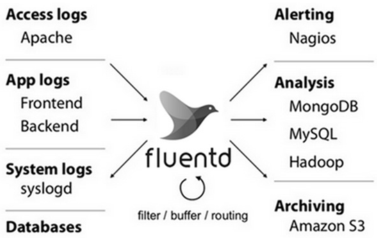


图 12-4　Fluentd 工作流程示意图

在提供日志告警分析服务的主机上也需要部署 Fluentd。这里的 Fluentd 进程用来接收从其他服务器上部署的 Fluentd 发来的日志。接收到日志后，Fluentd 可以根据配置再对日志进行过滤或转发给另一台 Fluentd 服务器，也可以直接保存在本地可访问的数据容器中，例如 Elasticsearch、MongoDB、Hadoop HDFS 等。

这种收集方式虽然方便，但是也有自己的问题。

第一个问题是不能加密。对于非 S3 的接收端 Fluentd，只能通过 HTTP 短连接进行传输，不支持 SSL 证书。[103] 如果担心这些日志会在网络上被其他人监听，那就要么自己定制，植入加/解密算法，要么使用其他支持加密的传输软件。

第二个问题是高可用机制复杂。这种问题不仅 Fluentd 会有，其他任何一个要完成异地数据传输的类似软件都面临同样的问题。因为网络和目标主机的可靠性肯定不是 100%，所以需要一个备用策略来转移传输失败的 HTTP 请求——这就是刚刚说的“高可用”（High Availability）。

2. 日志高可用

这一类问题有两种基本处理策略。

策略 1：日志最安全

日志安全的概念就是绝对不允许发生日志的丢失。只要日志在本地主机提交成功，就一定要保证日志在远程主机也提交成功，才算这条日志提交成功。而在确认这条日志在远程主机提交成功之前，本地主机必须处于“停等”的状态。这种方式看上去确实安全，能够保证本机和

100　官网地址：http://www.fluentd.org/。
101　官网地址：http://flume.apache.org/。
102　文档地址：http://hekad.readthedocs.io/。
103　目前的 Fluentd 版本默认只在 S3 为接收端的情况下提供对 SSL 证书的支持。

远程主机的日志高度一致，但是效率非常低。

策略 2：效率最高

效率最高的概念就是：只要日志在本地提交成功，就算提交成功，至于远程主机提交是否成功则根本不予理会。Fluentd 其实就是这种工作状态，它也只能在这种工作状态下工作。因为无论如何，这种将日志传递到远程的工作在 Fluentd 应用场景中是一个“增值服务”，不是本地主机业务方的业务目的，为了这个不太重要的目的牺牲大量的时间做“停等”是一种本末倒置的行为。

Oracle 数据库在 8i[104] 版本之后提供了一个叫作“Dataguard”的功能（如图 12-5 所示），其设计目的是实现高可用，尤其是像 Oracle 这种对实时性和事务都很重视的数据库产品更是如此。Dataguard 的工作原理是：用 1 台主数据库服务器 Primary 来接受正常的事务请求，再准备 1 台或多台Standby数据库服务器提供备用功能。在 Primary 上提交的 Redo 日志都会通过网络复制到 Standby 上，然后复用到 Standby 数据库中，从而使 Primary 数据库和 Standby 数据库中的数据保持一致。这样，一旦 Primary 数据库发生崩溃，可以立刻将对外服务数据库切换为 Standby 数据库。

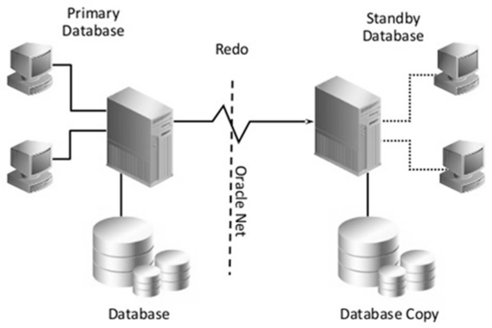


图 12-5 Oracle Dataguard 工作原理示意图

Dataguard 提供如下 3 种保护模式。

（1）最大保护

最大保护是指除非 Redo 日志在至少一个 Standby 上可用，否则事务不能提交。如果在某个 Standby 上不可用，则 Primary 数据库的操作会被停止。最大保护受到的制约比较多，在生产环境中并不常用——这就是前面说的“日志最安全”策略。

（2）最大可用

最大可用是指如果 Standby 数据库不可用，Primary 数据库仍然可以处理事务，只是在 Standby 数据库恢复后，Standby 和 Primary 数据库进行再同步。这样就导致了一个问题：在 Standby 数据库发生故障后到恢复前，如果 Primary 数据库也发生故障，就无法实现高可用了。如果强行设置 Standby 数据库为 Primary 数据库，有些数据可能会丢失。

（3）最大性能

最大性能是指 Primary 数据库的提交操作不等待 Standby 数据库。Primary 和 Standby 松耦合，

104 美国甲骨文公司的数据库拳头产品，8i 版本在 1998 年发布，目前最高版本为 12c。

数据保护级别较低——这就是前面说的“效率最高”策略。

所以，安全和效率这对矛盾的决策问题都会让人很苦恼。我们在实际生产环境中，只能在日志数据由于无法传递而产生的损失和为了避免这种损失而采取的技术方案所产生的费用之间做取舍。显然，不论选择哪一种，如果花费比损失小，这个方案就是一个基本划算的方案。

矛盾就是这样，没法两全其美，而一般情况下，通常也只能是牺牲实时性来保证数据完整性，或者牺牲数据完整性来保证实时性。只要选择的是后者，问题就来了——数据有一定的丢失概率，而我们可能要用丢失的数据来做后续的各种工作。这，只是不靠谱的开始。在这种情况下，我们能做的就是估算丢失数据的量（如果有必要）。估算方法有很多：一种方法是强行在主机上生成连续的日志编号（从 0 一直到 2^{64}），然后在服务器主机上统计一个时间段内收到日志的条目数与编号的最大/最小值的差值之商；还有一种方法，就是直接统计一个时间段内主机和服务器各自收到的日志条目数——都是用估算丢失量的方法来补充，对已收到的数据进行纠偏。

对于不太重要的日志，刚刚这些补救措施可以都不做，只要相信互联网和服务器的可靠性已经能够满足当前日志收集业务的需求就可以了——好在现在互联网业务的绝大多数场景都是这样。

12.2.2　延时上传

一些在终端使用 DTU 上传的场景中，终端通常没有存储数据的能力，所以这种情况通常不允许延时上传。这样，延时上传实际上只有如下两种场景了。

方式 1：在终端使用 Wi-Fi 或有线以太网上传

这种方式还是刚刚说的在类似智能手机或者 PC 使用的场景中上传。在这种情况下，设备的内存大都可以暂存数据，例如将收集到的数据积累 1 分钟或更久再上传，而不是随着扫描立刻上传。这种方式的好处主要是，使用短连接与服务器进行通信，可以减少开合服务器连接的次数，进而减少服务器端不必要的并发开销。

此外，这种方式还可以做一个增强，就是在网络失效的情况下积累更长时间的数据，直到网络恢复的时候再报送——让数据在本地“攒”一会儿。这种方式会在一定程度上降低由网络失效带来的数据报送损失，代价是本地终端要付出一定的空间。务必注意使用这种方案时的内存管理策略，不要影响正常软件功能的使用。

方式 2：在服务器端使用以太网上传

这种方式也是应用比较普遍的，尤其是在对实时性要求不强的场景中。在这种情况下，日志数据会以小时或者天为单位向指定的日志收集服务器上传。由于这种情况通常以文件作为传输单位，所以使用 HTTP 协议或者 FTP 协议传输较为合适。

12.2.3 加密问题

当终端或者服务器本地拿到数据要进行传输的时候，要注意数据加密的问题——是否担心传输的数据在网络上被人监听，然后被人恶意复制或篡改？

以太网的基本协议是 CSMA/CD（Carrier Sense Multiple Access with Collision Detection，带冲突检测的载波监听多路访问技术），也就是说，在一个子网中，一块无线网卡[105]在混杂模式下可以听到其他网卡传输的信息，这是极不安全的。在传输过程中，如果信息较为重要，一旦被他人监听，也有信息泄露的风险。所以，这种情况下通常采用加密传输的方式。

如果终端或服务器支持 HTTPS 协议栈，可以考虑直接使用 HTTPS 协议进行传输，代价就是每次传输连接建立的时候会花费更长的时间。如果终端或服务器使用 FTP 和 SSL 的文件传输方式，也相对比较安全。现在常用的 SSL 证书有 128 位、256 位、512 位、1024 位、2048 位（现在银行大多使用 2048 位的证书）等，位数越长，破解越困难，而且破解时间以指数级别增长。

在单片机和 DTU 的数据收集环境中，由于硬件资源所限，基本没有支持 HTTPS 协议的情况。如果要加密，需要自行编写（通常用 C 语言）加/解密算法。

这种加/解密按大类通常可分为对称加密和非对称加密。比较成熟的对称加密算法有 DES、3DES、TDEA、Blowfish、RC5、IDEA 等；比较成熟的非对称加密算法有 RSA、ElGamal 等。从安全性上来说，非对称算法要比对称算法更为安全。

1. 对称加密

DES 是 1972 年美国 IBM 公司研制的对称密码体制加密算法，该算法有 3 个参数。

$$f(key, data, \mathrm{mod}\, e)$$

其中，*key* 是一个 64 位的密钥，其中有 56 位参与加/解密运算，另外 8 位作为校验码；*data* 是要进行加/解密的数据串；mod *e* 是工作模式，有“Encrypt”（加密模式）和“Decrypt”（解密模式）两种。

如果密钥没有泄露，在信道上截获密文时，只能使用穷举法来破解。如果要全部穷举这些数据，假设每秒验算 100 万次，需要大约 2285 年才能完成。不过，现在大规模计算集群越来越普及，这种密码的可靠性实际上每年都在降低。因为密钥的副本不仅在服务器端有，在终端也要保存，所以，如果密钥在终端直接泄露，那就完全谈不上安全了。

2. 非对称加密

相比之下，RSA 比 DES 要安全许多。RSA 公钥加密算法是 1977 年由罗纳德・李维斯特（Ron Rivest）、阿迪・萨莫尔（Adi Shamir）和伦纳德・阿德曼（Leonard Adleman）一起提出的，

105 主要是指 IEEE 802.11x 协议族。

于 1987 年首次公布。当时他们都在麻省理工学院工作，“RSA”就是由他们三人姓氏的首字母组成的。

与 DES 不同，RSA 每次会同时生成一对密钥，即一个公钥（Public Key）和一个私钥（Private Key）。公钥是可以公开的，而私钥由生成方保存。RSA 加密的步骤如下。

（1）随意选择两个大的质数 p 和 q，p 不等于 q，然后计算 $N = pq$。

（2）令 $r = (p-1)(q-1)$。

（3）选择一个整数 e，满足 $e < r$，使 e 与 r 互质。再求 d，使 $ed \bmod r = 1$。

（4）销毁 p 和 q。

这时得到的 (N, e) 就是公钥，(N, d) 就是私钥。

如果要加密，可以把每个字符的 unicode 码 u 做以下变换：

$$u^e \bmod N = c$$

然后，把每个 c 都传给解密方。

解密方得到 c，要想得到 u，则有

$$c^d \bmod N = u$$

由于这些都是正整数，所以对只会做加法的电子计算机来说，还是能比较容易地得到解。

但是，要在破解公钥（本来就是公开的）的情况下拿到密文 c 来破解 u，就只能根据

$$de \bmod (p-1)(q-1) = 1$$

来计算，也就是通过截获的 N 来分解质因数，得到 p 和 q。

这个计算量在 512 位的 RSA 密钥上大概需要计算 8000 个 MIPS[106] 年才能破解。而随着电子计算机计算能力增加，在加密时只需要加长 RSA 密钥的位数就可以了。

一般来说，不论加密还是解密，RSA 的运算成本比 DES 要高。所以，必须使用 RSA 这种级别算法的加密场合在对保密性要求不高的情况下很少见。

12.2.4　压缩问题

在日志收集中还有一个问题需要考虑，就是数据压缩问题。

在互联网环境中，带宽的价格是比较昂贵的，如果 CPU 资源相对过剩，而带宽资源又比较紧张的话，可以考虑压缩后再将数据上传。如果上传的是日志文件，可以直接使用压缩软件或者自己编写的压缩算法进行压缩，然后通过 FTP 上传。

如果使用的是单条的日志数据，那么一般只能在自己编写的算法里实现。如果使用 Python 语言在客户端进行日志收集的话，可选择的库非常丰富。zlib 就是 Python 语言环境中很常用的

106　“MIPS”是“Million Instructions Per Second”（单字长定点指令平均执行速度）的缩写，每秒处理的百万级的机器语言指令数。

一个用于压缩和解压缩的库文件。

下面给出一段使用 zlib 进行压缩和解压缩的简单语句，由于通常是基于字典压缩，所以重复字符串越长，所占全文比例越大，压缩的效果就越好。凡是基于字典压缩，通常都会有这种经验——文件形式的内容比单条形式的内容压缩率更低，压缩效果更好。这段文字压缩前是 228 个字符，压缩后仅为 26 个字符。

```
#coding=utf-8
import zlib
source = "xxxxxxxxxxxxxxxxxxxx,xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx,xxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxx,xxxxxxxxxxxxxxxxxxxxxxx,xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxx,xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
compressed = zlib.compress(source)
decompressed =  zlib.decompress(compressed)
```

请注意，这里的压缩一般情况下都是无损压缩，所以压缩的数据肯定不会低于信息论的下界，也就是说，对“稀疏矩阵”的压缩效果尤为明显。我们看到的现象就是，如果文件中有大量重复信息，那么压缩出来的文件就非常小；而如果文件是被压缩过的，例如 .JPG 文件、.MPEG 文件等，则不管使用什么算法，压缩空间通常都很小。

12.2.5　连接方式

在数据采集这个环节，还有一个问题需要讨论，那就是使用短连接还是长连接。这两种方式在互联网的应用中都广泛存在。

1. 短连接

先说短连接（如图 12-6 所示）。

在 TCP 协议栈中，连接的建立有标准的步骤，就是 TCP 协议栈规定的“三次握手”。这里有两个角色：一个是 Server，它是被动方，所做的事情就是启动一个 TCP 端口的监听；另一个是 Client，用于主动向服务方 Server 发起连接请求。

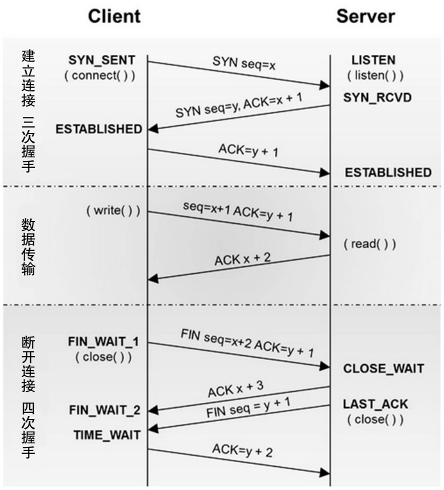


图 12-6　三次握手与四次挥手

（1）当 Server 接收到 Client 发来的连接请求（SYN 包）后，会回发一个 SYN 包，表示 Server 方已经准备好建立连接。

（2）在 Client 接收到 Server 发来的 SYN 包后，回送给 Server 端一个 ACK 包。这样，三次握手就建立了一个五元组连接，在 Server 端和 Client 端都会看到一个新五元组的建立。这个所

谓的五元组是指“本地 IP”、“本地 Port”、“远端 IP”、“远端 Port”、协议类型”的组合，用来描述和管理一个独一无二的连接。有些文章中会使用“源地址”、“远端口”、“目的地址”、“目的端口”这样的说法，意思相同。

（3）当连接建立以后，Client 通常会直接把要传递的数据通过这个五元组描述的连接传输给 Server，传输完成后关闭连接。

关闭的过程叫作“四次挥手”，过程如下。

（1）Client 对服务器发送 FIN 包，表示 Client 不再有任何数据要传递给 Server。

（2）Server 对 Client 回复 ACK 包，表示 Server 已经收到了 Client 发来的 FIN 包，但此时 Server 还可以通过这个连接向 Client 发送数据。

（3）Server 对 Client 回复 FIN 包，表示 Server 没有数据要传递给 Client 了。这个时候，作为线上服务器，Server 通常可以在没有收到 Client 最后发来的 FIN 包的情况下直接关闭连接，回收连接所占用的内存资源。

（4）Client 对 Server 回答 ACK 包，表示 Client 收到了 Server 最后发来的 FIN 包，正式关闭连接。

之所以称为短连接，是因为这样的过程耗时极短。通常在手机、PC 终端发起一次 HTTP 请求，报送数据的量以几百字节（B）到几十千字节（KB）居多，所以，从建立连接，到发送数据，再到连接断开，大概要经历几十毫秒到几秒不等，主要开销是在网络上传输的延迟。这种短连接的报送方式最为普遍，网页上常用的 GET 方式和 POST 方式的请求就属于短连接。这种连接适用于高并发的短小访问，例如单条日志的报送。

2. 长连接

顾名思义，长连接的连接时间较长。它和短连接的区别在于，在三次握手之后，通常要进行不断往复的数据交互，即在连接建立后，Server 和 Client 之间会一直保持着这个连接，并通过这个连接相互发送数据。

从功能上来说，长连接的好处是连接断开之前 Server 能在任意时刻向 Client 发送数据信息，这个功能是短连接不具备的，好处显而易见。而且，再次从 Client 向 Server 发送消息也不用进行三次握手。我们平时用的 QQ、微信等各种 IM 工具都要至少拥有一条基于长连接的连接来保持信息往返的畅通。

当然，长连接也有它的缺点，就是对 Server 和 Client 的内存都有一定的占用——因为要维护这条连接是需要内存保存信息的。而对于可以同时容纳数十万长连接的 Server 来说，内存占用当然就更为明显。从工程经验角度来看，大部分服务器上的轻传输业务的长连接通常只占用 1KB 到 4KB 不等的内存，具体占用的大小与内核设置有关，也同时与长连接上传输的内容多寡有关。具体占用的情况可以在实测时用 free 命令（Linux 命令）查阅。

12.2.6 消息格式

对于一种已经定制好的日志来说，把日志信息原原本本报送给服务器就可以了，通常不需要进行太多其他处理。但是，如果这种数据收集具有比较大的自主性，甚至在收集数据的生命周期中还需要进行不定时的修改，这个时候就要下点功夫设计整个数据收集的逻辑了。按照我的经验，一般需要注意两点。

注意点 1：报送格式尽量采用 Key-Value 键值对的形式

这种形式的好处是格式清晰，可扩展性高，Key 和 Value 一一对应，在报送到服务器的时候方便进行切割和对应归档。而且，如果需要进行删减或者增补，只要直接删减或者增加键值对就可以了。

在处理 HTTP Web 请求时使用的 GET 方式就是一种非常典型且非常好用的键值对方式。以“https://www.baidu.com/s?wd=vgg19&rsv_spt=1&rsv_iqid=0xf5ddc0c10008f89b&issp=1&f=8&rsv_bp=0&rsv_idx=2&ie=utf-8&tn=baiduhome_pg&rsv_enter=1&rsv_sug3=6&rsv_sug1=5&rsv_sug7=100”这个 URL 为例，“https://www.baidu.com/s”是服务器的地址和虚拟路径，“?”表示其后都是 GET 方式的键值对。这些键值对之间用“&”分割，键值对内部用“=”分割，内容清晰。Web 服务器只要对这个 URL 进行解析，就可以把所有要报送的值都接收下来。

```
wd=vgg19&
rsv_spt=1&rsv_iqid=0xf5ddc0c10008f89b&
issp=1&
f=8&
rsv_bp=0&
rsv_idx=2&
ie=utf-8&
tn=baiduhome_pg&
rsv_enter=1&
rsv_sug3=6&
rsv_sug1=5&
rsv_sug7=100
```

使用 Web 服务器接收数据时也可以采用 POST 方式，其接收方式与 GET 类似，但发送的数据量更大，在此就不赘述了。只补充一句：在这个环节，不论是 GET 方式还是 POST 方式，都有一些传输非 ASCII 字符的场景。在这些场景中，会考虑使用 Base64 编码进行传输，然后在接收端进行解码（如图 12-7 所示）。Base64 的编/解码方式是开源的，在 Java、Python 等语言中都有很成熟的实现类。Base64 编码不是用来加密的，而是用来实现二进制到文本的传输。

注意点 2：自举要方便

所谓“自举要方便”，是指在终端采取的数据收集策略要比较灵活，要允许在几乎任意时刻

对收集策略进行修改。这种方式很容易实现，可以从两个地方入手，一个是终端自身的自举程序，另一个是策略文件。

自举程序像 PC 硬件的自举一样，有一种约定的标准启动方式。

PC 的自举过程是：在启动时先读 BIOS，再按顺序读取 BIOS 中“1st Boot Device”、“2nd Boot Device”、“3rd Boot Device”这些设备的启动扇区，通过启动扇区里面的程序指针完成一系列自举的程序段跳转，最后引导计算机系统启动（如图 12-8 所示）。如果没有这样一个“雷打不动”的约定步骤，计算机系统是不可能启动的。

图 12-7　Base64 编码

图 12-8　PC BIOS 中的 Boot Device 配置

在每台终端上安装数据收集程序也是一样的。数据收集程序通常要做两件事，第一件事是周期性地前往一个 URL 地址下载策略文件，第二件事是确定执行这个策略文件的内容。

所谓“周期性地前往一个 URL 地址下载策略文件”，是指程序里面要有这样的功能：在每天、每小时或者更短的时间间隔，到一个 URL 去下载一个策略文件。这个周期可以按照自己的业务需求而定。如果周期太短，不仅会给服务器造成不必要的压力，也没有必要；如果周期太长，策略生效的实时性也会降低。所以，对这个度要好好考量，并且要紧扣业务需要。文件的 URL 最好是一个域名，这样就不会因为 IP 地址的迁移而导致地址失效。

策略文件就是具体的报送逻辑——需要报送什么字段，以什么逻辑收集，报送到哪里，用什么格式承载这些信息。一般的策略文件都是脚本文件，由终端上的脚本库提供支持。在智能手机上，通常通过 App 版本的迭代直接升级（用 C 或 Java 编译的二进制文件居多）。PC 上的情况类似，在程序里加入一些升级模块，可以直接升级一个 .dll 文件，或者用其他库文件来更新。如果终端环境相对封闭（例如专用的 ARM 平台），也可以考虑在其中加入 Python 或者 lua 库，这样就可以将 Python 或者 lua 脚本作为策略文件，可读性更强，版本管理也比较方便。

12.2.7　维度分解

维度分解是日志收集中在入库前的最后一个环节。所谓维度分解，就是把收集上来的消息格式分解成数据库或数据仓库可以识别并存储的格式（如图 12-9 所示）。

在收到一条消息后，根据分隔符进行切割，把原来的一串字符切分成不同含义指代的维度信息。

在使用标准的 GET 或 POST 方式时，通常可以直接使用第三方提供的类（Class）进行拆分，把收到的信息转换成一个数组或哈希表，这也同样是一种 Key-Value 对的形式。

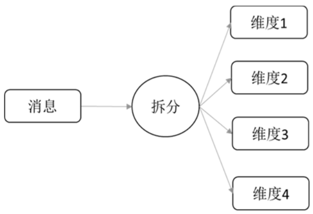


图 12-9　维度分解

如果是自定义的字符串格式，而且字符串中表示了多个不同的含义，需注意分割不要产生歧义，分隔符不应出现在维度的 Key 和 Value 中。所以，在自定义字符串格式的时候，推荐使用 Base64 对 Key 和 Value 进行编码。由于在 Base64 传输的字符中，只能由 A ~ Z、a ~ z、0 ~ 9、“+” 和 “/” 这 64 个字符和 “=” 作为正文内容，所以只要使用这些以外的字符作为分隔符，就可以保证信息不失真。这种方式的代价是编码后数据会比原始长度长约 $\frac{1}{3}$（这是工程经验值，在实际应用中仍然需要具体测算）。

在维度分解后进行保存时，建议保存一份原始消息数据作为存档信息，在类似 HDFS 这种可低成本扩容的分布式文件系统上进行压缩存储，在 Hive、Oracle、MySQL 或 HBase 等数据库（数据仓库）中保存一份分解后的维度与值的描述信息供查询与统计使用——前者供出现问题时进行倒查，后者供日常业务运营使用。

如果报送的字段内容不确定，而且随时可能扩展或者取消，尤其是在互联网这种变化极快的产品环境中，还是建议将它们放入类似 Cassandra 或 HBase 这样的 NoSQL 数据库中，不仅使用比较方便，更改的代价也比较小。

12.3　这只是不靠谱的开始

在一个复杂耦合的系统中，任何一个独立组件的有效性都会影响整个系统的有效性。就拿数据采集系统来说，每个采集点的有效性 $P_{collect}$、网络传输的可靠性 P_{trans}、服务器的有效性 P_{server} 都会影响最终收集到的数据的有效性（或称“可靠性”）。如果没有办法直接从提供商那里得到这 3 个数值，就只能用其他办法进行测算了。

我们举个例子，假设能够测算出这 3 个值。

在测算过程中发现，$P_{collect}$ 的有效性为 99.9%，即平均 1000 条信息报送中会有 1 条信息由于收集终端的问题无法报送成功。

P_{trans} 有 1% 的失效概率（这个值已经很高了，说明运营商的网络状况非常不理想），那么 P_{trans} 为 99%。这里的失效概率不是丢包率，因为使用 TCP 协议族（包括 HTTP 和 HTTPS）是有重传机制的，可以理解为网络层的故障、运营商的网络问题等导致在终端和服务器都正常的状态下无法访问的情况。

P_{server}有 0.1% 的失效概率，即一段时间（例如 1 年）里有 0.1% 的时间处于无法工作的状态，而且这个时候也无法进行数据保存和重传。

$$P_{collect} = 0.999，P_{trans} = 0.99，P_{server} = 0.999$$

最终服务器上收集的数据量在实际产生的数据量中所占的比例，就可以用下面这个公式来计算：

$$P = P_{collect} \times P_{trans} \times P_{server}$$

$$P = 0.98802099$$

通过计算，数据收集率约为 98.8%。如果这个指标已经满足了目前的业务需求且成本合理，就可以大胆使用了。如果无法满足需求，就要看可以在哪些环节进行优化，优先优化投入少且改进明显的环节。

其实，不管怎么优化，都面临投入成本和改造收益的权衡问题。刚刚说的这个场景在实时收集的场景中效果比较明显。在延时收集场景中，由于有足够的时间缓冲，可以等待网络的恢复或者终端、服务器等硬件的修复，所以只要数据保存策略没有问题，P 的值可以无限接近 100%。

12.4　本章小结

数据采集是数据生命周期的开始，充分考虑各个环节能给整个大数据系统带来非常关键的影响。

对于高昂的带宽资源，文本压缩是一种比较好的选择。例如，把文本日志进行压缩后传输，通常可以有效地解决带宽不足的问题。是否一定要传输原始日志，是否可以进行有损压缩，在具体应用场景中都是可以探讨的问题。

此外，由于加/解密本身也需要成本，所以在选择方案的时候要评估成本及其带来的收益。如果有些数据确实没有太高的保密价值，就不要做加密了。而在确实需要加密的场合，加密的方案也要与时俱进，因为破解能力也会随着计算机计算能力的增强而提升。在任何时候，只要能保证当时的破解成本远高于破解后的收益，就是一种性价比合适的加密方案。

在数据采集环节，除了由 DTU 或者终端设备直接报送的数据，还有很多由分散在各处的工作人员观察并记录下来的数据。我曾经在某连锁便利店中见过，负责结算的工作人员会观察当前进行交易的顾客的年龄（当然只是粗略估计年龄段），并通过收银系统将此数据记录在案。

获取数据的环节很多。在获取数据的过程中，非常需要用“匠人精神”去把控每一个细节，并通过工具改造、环境改造等方式逐步进行完善和优化。

第 13 章　数据存储

当一台服务器收到由其他服务器传来的日志数据时，一般有两种处理方式：一种是在内存中直接处理，进行流计算，计算完毕后将计算结果保存或者输出到指定位置，原始日志数据直接丢弃；另一种是直接写入持久化存储介质（例如文件、数据库等），以备后续的查询和统计分析等。还有一些场景会将两种方式结合使用，既做流处理用于快速反馈数据，也把日志进行持久化归档处理。

流处理不属于数据存储的范畴，这里就不多讨论了。我们只需要理解这样一个概念：这种处理就是在内存中留一个池空间，当数据从网络高速写入这个内存池空间的时候，可以根据需要把这个池空间当成磁盘上的文件进行相应的统计计算，然后设置清空策略（可以采用滑动窗口策略，也可以进行固定时长清空）。

将数据保存到磁盘上是非常讲究的，需要考虑的问题也比较多。对一台服务器来说，当它收到数据准备写入磁盘的时候，会面临很多选择——没错，还是各种矛盾的权衡。

13.1　读写不对等

一台服务器的硬件资源确定后，对于存储就有这样一个问题需要考虑——读写场景是什么？

“读写都少”的情况基本不用考虑对矛盾的权衡，因为此时的磁盘 I/O 资源非常充分。而在“读多写少”、“读少写多”、“读写都多”这 3 种场景中，还是要考虑资源怎么分配才能最大化地满足业务需要。

另外，读写操作涉及的范围是什么形态（I/O 模型）也很重要。就拿读来说，读得频繁而每次只读取少量数据是一种情况，即“小 I/O 频繁读”；读得不频繁而每次需要读取大量数据是另一种情况，即“大 I/O 低频读”。更有甚者，读得频繁且每次都读取大量数据，这种场合下往往都会使总吞吐量超过磁盘 I/O 的吞吐上限，处理原则通常是先设法将这个业务分散到多台物理机上，也就是化简这种情况，使之变成前两种情况中的一种再做考虑。遇到任何一种情况都无须惊慌，因为对这些情况都有成熟的解决方案。

凡是最后要存储在磁盘上的数据，都要使用文件系统作为落地方式。即使是像 Oracle 裸设

备或者 Oracle ASM[107] 这种不使用 Linux 系统作为文件分配表的系统，也不过是 Oracle 自己接管了磁盘，自己编写文件分配算法而已。以 CentOS 为例，CentOS 6 使用 ext4 作为默认文件系统，CentOS 7 则改用 XFS 作为默认文件分配系统。对于文件系统的选择，我们一般关注读写吞吐速度、可靠性、可扩展性这 3 点。

对于磁盘 I/O 已经确定的情况（传统的机械硬盘取决于磁盘自身接口带宽、主板 SATA 或 SAS 总线带宽及磁盘转速等因素），改进吞吐能力的空间通常很小。

13.1.1 读多写少

“读多写少”是一种在互联网领域很常见的读写模型（如图 13-1 所示）。在微博、新闻等网站都会有这样的模式：一个进程向磁盘上的某个地方写了一段数据，无论数据载体是数据库还是文件，此后高频次的访问都会扫描磁盘的这块地方。

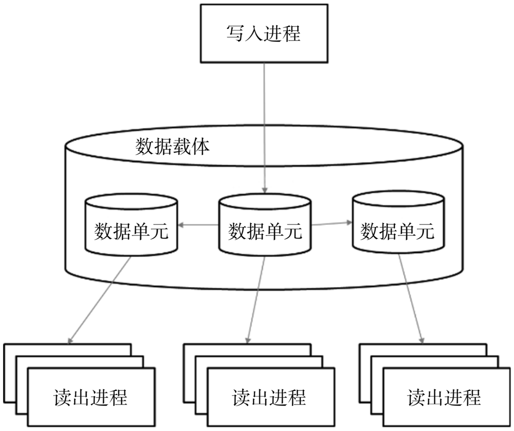


图 13-1 读多写少

这种情况下，通常考虑直接把读 I/O 的部分分摊到多个读 I/O 单元中去。例如，MySQL 的读写分离就是一种典型的应对“读多写少”的方式，先在主库中写一份数据，再同步到 N 个从库。这样，再从从库中读取数据时，吞吐能力就变成了原来的 N 倍。这种方式在其他类似的场景中同样适用。

13.1.2 读少写多

“读少写多”是一种在数据收集过程中比较容易见到的场景（如图 13-2 所示）。

当多个数据点向一个数据载体写入数据时，伴随着相对较少的数据读取。对于较高的写并发，同样建议把写并发的 I/O 分散到不同的主机或者磁盘上，这种情况下使用 Hadoop 或者 Ceph

107 Oracle 自动存储管理（Automatic Storage Management）。

一类的集群均可。在 Hadoop 集群中，同样建议将较小的数据积累成较大的数据块（例如数百 MB 的数据块）再写入，这样做效率比较高。因为每次写操作都要对 NameNode 这个热节点进行访问，所以还是要尽可能减少这种访问带来的单点压力。

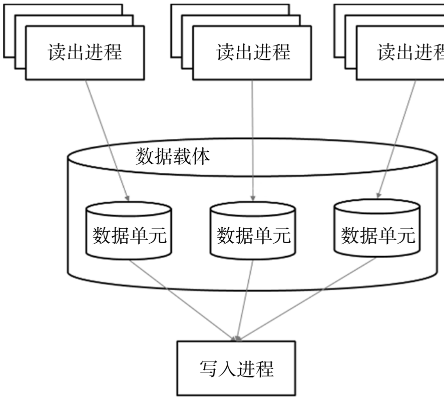


图 13-2　读少写多

13.1.3　读写都多

对于“读写都多”的场景，除了在单台计算机上使用诸如 RAID10 这样的技术进行数据条带化（Striping）以外，还可以利用集群读写分散的特性，或者使集群中各台主机的读写时间错峰的特性（如图 13-3 所示）。

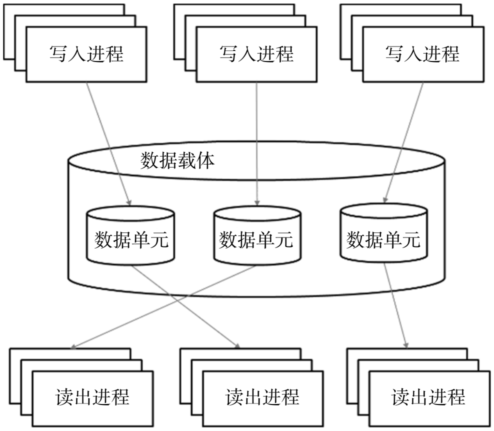


图 13-3　读写都多

在这类场景中，使用 Hadoop（HDFS）或者 Ceph 一类的集群存储系统当然可以，但一定要尽量避免将读写操作都放在某一台或几台计算机上，进而导致性能歪斜。我的建议是先看读出

的特性。以 Hadoop 集群为例，如果读出时每次都扫描时间最晚的写入数据块，就有可能与写入进程同时访问相应的主机。这种情况下，建议使用与时间无关的哈希方法来写入不同的文件，从而保证将数据写入不同的 DataNode 节点。如果读出是半随机方式的，没有明显的时间偏好，写入时就可以不用太在意这一点，只要按时间顺序将一个文件写入 Hadoop 集群就可以了。

解决 I/O 征用的方法其实比较少，主要就是靠征用资源的时间和空间的分散化，这是最为根本的思路。

13.2　进快还是出快

在第 11 章中我们学习了很多关于排序和检索的算法，其中有一个体会：由于数据的产生大部分是随机的，所以，如果想在某一次检索中很快地检索出一个数据，那么在检索之前准备工作的成本就比较高；相反，如果不想花费这个准备成本，就要忍受在检索时所花费的成本。这就是一对矛盾。

我们先来说说追求写入速度的情况。

13.2.1　最快写入

在选定一块 SSD 或者机械硬盘的时候，I/O 峰值是有限定的。所以，这时写入的最高速度也不可能逾越硬盘自身的写入峰值。以我使用的测试机的硬盘为例，可以用“dd”命令进行写入操作，以测试连续写入速度。

```
root@henry-B85-HD3-A:/tmp# time dd if=/dev/zero of=/tmp/writetest bs=100M count=10
10+0 records in
10+0 records out
1048576000 bytes (1.0 GB) copied, 0.689278 s, 1.5 GB/s

real 0m0.690s
user 0m0.000s
sys  0m0.484s
```

通常第一次测试的速度都不太“稳定”，会显得写入速度非常快。例如，在以上测试中我们可以看到，写入速度大约为 1.5GB/s，但实际上不可能达到这个速度，因为大部分数据都写到了缓冲区中。这种情况下，逐步提高写入的数据量就可以测出磁盘真实的写入速度了。

```
root@henry-B85-HD3-A:/tmp# time dd if=/dev/zero of=/tmp/writetest bs=100M count=20
20+0 records in
20+0 records out
2097152000 bytes (2.1 GB) copied, 11.3685 s, 184 MB/s

real 0m11.476s
```

```
user 0m0.000s
sys  0m1.240s
root@henry-B85-HD3-A:/tmp# time dd if=/dev/zero of=/tmp/writetest bs=100M count=30
30+0 records in
30+0 records out
3145728000 bytes (3.1 GB) copied, 16.4094 s, 192 MB/s

real 0m16.595s
user 0m0.000s
sys  0m1.884s
root@henry-B85-HD3-A:/tmp# time dd if=/dev/zero of=/tmp/writetest bs=100M count=40
40+0 records in
40+0 records out
4194304000 bytes (4.2 GB) copied, 21.6509 s, 194 MB/s

real 0m21.906s
user 0m0.000s
sys  0m2.552s
root@henry-B85-HD3-A:/tmp# time dd if=/dev/zero of=/tmp/writetest bs=100M count=50
50+0 records in
50+0 records out
5242880000 bytes (5.2 GB) copied, 27.1801 s, 193 MB/s

real 0m27.637s
user 0m0.000s
sys  0m3.360s
```

这个稳定在 190MB/s 的写入速度就是当前这块 SSD 磁盘的写入峰值速度。

机械硬盘（如图 13-4 所示）的测试方法与此类似。在盘片高速旋转的过程中，机械硬盘的磁头根据 CPU 传来的指令滑动，同时进行读写操作。对于机械硬盘，写入速度最快的方式就是在当前文件的尾部追加数据，这属于连续写入。因为所有的数据在磁盘上是连续的，也就意味着可以在高速旋转中不间断地写入数据，没有过多的寻址开销，所以对硬盘来说这种写入方式效率最高。如果这些数据是按照时间顺序直接来源于外界报送的话，这就是一种按照时间顺序连续写入的场景。

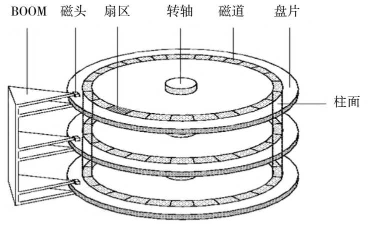


图 13-4　机械硬盘结构

但是，这种写入方式除了在需要连续读出整个文件的场景，在其他场景中效率都很低。例如，要在这样一个文件中找到包含“hello world”的信息，除了把整个文件全部读到内存里进行一次过滤，别无他法。这种查找的时间复杂度，与在一个单向链表中用穷举的方法找到一条数据相同。

$$\text{查找次数：}\frac{n(n+1)}{2} \qquad \text{时间复杂度：}O(n^2)$$

现在越来越多的数据收集场景中不再直接使用文本文件进行数据保存的原因也在这里。如果使用文本文件，那就纯粹是做压缩归档保存，以备未来进行审计或深度分析时使用。

13.2.2　读出最快

如果要让文件以最快的速度读出，就需要在文件写入磁盘之前进行排序处理。最常见的方式应该是在文件内容层面建立索引，并实时维护索引文件。这个过程非常复杂，非专业人士通常不会选择用这种方式来直接操作文件。更好的方式是使用数据库中的相应功能来实现。

追求“读出快”是一个比追求“写入快”更复杂的课题，因为这取决于读出时的场景。如果按照时间顺序连续写入，且按照时间顺序连续读出，这就是输入非常理想的、读写速度都非常快的情况。平心而论，这种情况太理想了，非常少见。在按照特定的顺序进行连续读取或随机读取的场景中，需要做强排序。Oracle 中的索引组织表（Index Organized Table，IOT）就是其中的一种（如图 13–5 所示）。

这种表很特殊，在构建时要先声明表的类型为索引组织表，每次插入数据时，整条数据会被直接插入索引表。索引结构在这里既充当了索引结构，又充当了表数据载体。所以，其数据扫描与普通索引相比，不需要通过索引访问 ROWID，再跳转到堆组织表。这种组织结构的好处是，只要是根据索引键连续扫描（索引键 BETWEEN a AND b），读出速度都非常快。当然，单条扫描的速度也非常快，只是与普通的索引项相比没有优势（除了在开始插入时需要不断调整这棵“索引树”以外，索引结构的其他劣势并不明显）。关于索引的组织方式，13.5.2 节会专门讨论。

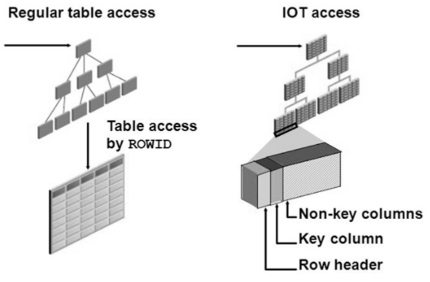


图 13–5　堆组织表和索引组织表

13.3　文件还是数据库

在 13.2 节我们已经讨论过，如果数据库中的数据是供使用者频繁查找的，尤其是扫描所获取的内容在被扫描的数据量中占比很低的情况下，通常不推荐写入程序直接写文件，并把该文件作为查找介质直接使用。在这种场景中，首选肯定是数据库，其数学方面的原理会在 13.5.2 节讨论。

数据库作为专业的数据存储和查找引擎，已经有超过 30 年的历史了，而且其功能越来越强大。以甲骨文著名的数据库产品 Oracle 为例，其数据库引擎 Oracle Database 从 8i、9i、10g、11g

到 12c，一路升级，辅助性组件也在不断丰富。与 Oracle Database 结合最为紧密的有 Dataguard 和 Golden Gate 等补充型产品组件。Dataguard 提供 Oracle Database 高可用解决方案，Golden Gate 则用来做数据同步。

数据库作为数据存储的载体，其选型也是有一些讲究的。

从数据的组织形式来说，目前有两大类，一类是 RDBMS（关系型数据库管理系统，Relational Database Management System），另一类是 NoSQL[108]。

RDBMS 通常用数据库、表、字段 3 层对数据进行描述。这种结构应用非常广泛，历史悠久，普及性极高。在这类数据库中，一个数据库（Database）下可以建立若干个表（Table），一个表中可以建立若干个字段（Field）。在一个表的内部，每一行数据都必须具有相同的字段描述。

NoSQL 通常是用数据库、表、键值对 3 层对数据进行描述。一个数据库下可以建立若干个表。NoSQL 和 RDBMS 的区别在于，NoSQL 表内部不设置字段，表内部的每一行数据都用 Key-Value 键值对来描述，也就是说，每一行的字段都可以不同。

乍一看这只是形式上不同，其功能都是存储数据，但是它们的使用场景有着比较大的区别。RDBMS 注重关系，并且在查找引擎上注重对多表联查[109]的支持和优化，适合做复杂的多维数据分析。而 NoSQL 表天生就是一种稀疏矩阵的数据逻辑架构，适用于那种在大量不规则的数据结构中查找一条数据的场景，例如博客系统等。由于多表查询很不方便，所以在供分析的数据存储场景中，通常不推荐使用 NoSQL 类型的数据库。

在数据量非常大且预算有限的情况下，还是推荐使用开源的 Hive、SparkSQL 或者 PrestoDB on HDFS。这类数据仓库对海量数据的支持特性非常优秀，而且有丰富的文档可以参考。

13.4 要不要支持事务

对于在线数据库 OLTP 来说，事务通常是比较重要的。事务是指那种一次涉及多个表、多条数据的写操作[110]，要求是要么都生效，要么都不生效。

事务在对业务提供线上支持的数据库中是非常重要的，尤其是对涉及支付的系统，更是必需的。不过，我们所说的日志分析场景中的数据不一定是必需的，而且在我的工作经验中，没有事务的情况似乎更好、更自由。

在数据分析场景中，通常会使用 OLAP 数据仓库。常见的 OLAP 数据仓库有 Oracle、DB2、Teradata、Greenplum 等。这些数据仓库的特点是吞吐量比较大，尤其是对多表联查的效率支持很高，而其缺点也非常明显，例如对一次涉及几条数据的小事务来说性能支持非常不好。

---

108 NoSQL 是指非关系型数据库。

109 基于笛卡尔乘积（Cartesian Product）的查询与过滤。

110 包括插入（Insert）、更新（Update）和删除（Delete）。

如果觉得上述这些数据仓库价格高昂，还有 Hive、SparkSQL、PrestoDB on HDFS 可以选择。这些分布式数据仓库通常有更高的单次访问延迟，不过这种代价对于 OLAP 的场景来说无足轻重，关键是获得了极高的可扩展性和大规模数据的读写性能。这个性能的提升主要通过“读写分散”获得，下面我们就以 Hive 为范例进行讲解。

Hive 在底层是基于 HDFS 和 MapReduce 的，写入集群时遵循 HDFS 的写入原则。只要向 Hive 的数据仓库里写数据，就必须使用诸如

```
LOAD DATA LOCAL INPATH '/Linklog_20161122.txt' OVERWRITE INTO TABLE Linklog PARTITION (dt='20161122');
```

这种语句。

以上语句实际上就是把本地文件“/Linklog_20161122.txt”复制到集群 hdfs:///user/hive/warehouse 中去。这里为什么不使用事务呢？因为在 OLAP 环境中，数据一般都是已经存在的历史数据，几乎没有回退的机会。如果一定要使用事务，就必须有与 Oracle 结构相同的更新逻辑。

以更新一个数据块 D 为例，如图 13-6 所示。

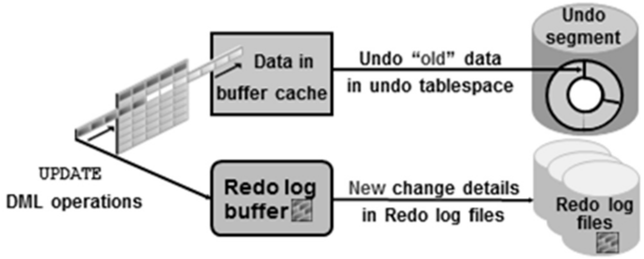


图 13-6　事务中的 Undo 和 Redo

（1）要更新数据块 D 的时候，为了保证事务隔离，首先必须保证其他事务能访问提交前的数据块 D，此时会生成一个 Undo 块信息 U 来记录数据块 D 更新前的信息。

（2）生成一个 Redo 信息来记录 Undo 块涉及的修改内容，包括 U 的地址、U 的内容、事务标识（事务号）。

（3）修改数据块 D 的内容，同时产生一个 Redo 信息，包括数据块 D 的地址、数据块 D 修改后的内容、事务标识。

如果此时用户提交了 Commit 命令，那么 Undo 块信息 U 的内容就会被释放，事务将无法回滚。如果此时提交了 Rollback 命令，那么数据块 D 的内容会被 Undo 块信息 U 的内容替换。这两种情况都标志着这次事务操作的结束。而在事务操作结束之前，任何涉及数据块 D 的读操作都会命中 Undo 块信息 U，而不是数据块 D。这种方式能够在事务的一致性、回滚速度等方面实现极好的平衡。不过我们也可以清楚地看到，一个数据块的 I/O 操作实际上产生了若干个数据

块的 I/O 操作，而且这里还没有讨论数据块 D 涉及的索引块对应的改动操作及锁机制，这些操作对一个 OLAP 环境来说基本上是不需要的。如果发生了，就意味着极大的 I/O 开销。OLAP 环境中的更新操作一般情况下都是整片数据的删改问题，所以完全没有必要做基于记录层的事务，直接使用文件会使开销更小。

在 Hadoop 的 HDFS 中写一个文件（就像刚刚用 HiveQL 做 LOAD 命令那样），会引发对一个文件的复制操作。这个过程是一个轻型的复制过程，如图 13–7 所示。

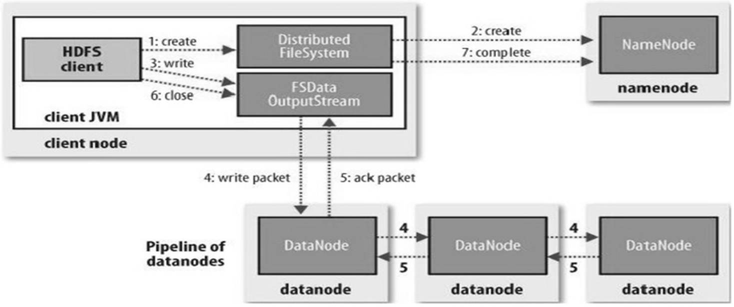


图 13–7 Hadoop HDFS 写文件流程

（1）客户端会向 NameNode 发出一个写入请求。

（2）客户端直接向某一台 DataNode 写入文件块。

（3）DataNode 会根据集群配置中的 Replication 数量做 DataNode 之间的文件块同步。

（4）写完后，客户端会关闭写连接，并向 NameNode 同步一个写完毕的信息。

这种流程的好处显而易见——中心压力小，写分散，单次写入信息多，从而减少了平摊写入开销。所以，基于 HDFS 的 Hive 成为一种永久存储环境下性价比很高的解决方案。

13.5 表分区和索引

为了提高检索效率，有时需要建立表分区和索引。

索引大家应该不会陌生，在 OLTP 环境中就经常使用它。索引的好处是检索的时候能够有效避开很多不必要的数据块扫描，坏处是在进行数据插入、更新、删除的时候需要重新对索引进行维护，这是一笔时间开销。常见的索引有 B–Tree 索引、Hash 索引、Bitmap 索引，它们都能在不同的场合加快检索速度，但是通常也有各自的局限性。

在 OLTP 环境中，只要索引设计合理，通常都能够加快检索速度，同时降低一定的更新速度。而在 OLAP 环境中，在实践过程中我们可能会感受到，在很多场合，索引对检索的加速效果不明显。

13.5.1　表分区

表分区在 OLAP 环境中通常会建立在时间字段上（也可以建立在其他字段上），将这个字段作为分区键（Partitioning Key）。如果分区键是有序类型的，通常 SQL 查询语句会在一个 where 谓词对分区键作出取值限制后忽略扫描不涉及的分区，在查询 SQL 语句没有做任何特殊优化的情况下，把查询效率提高 n 倍（n 为分区个数，且分区尺寸大小相近）。

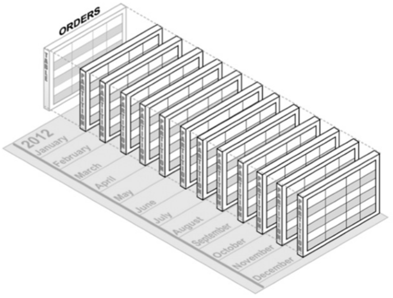


图 13-8　表分区示意图

如图 13-8 所示，如果在一个历史数据表中按照年和月来划分分区，那么当一个 SQL 语句指定为

```
SELECT ……
WHERE DATETIME='2012-12';
```

时，扫描内容会限定在“2012-12”这个分区中，其余的分区表及其中的索引不会被扫描。

当然，如果分区非常不均匀，例如“2012-12”分区中的字段有 100 万条记录，而“2012-1”至“2012-11”分区中各有 1 万条记录，则效率的提升理论上仅仅是从扫描 111 万条数据减少到扫描 100 万条数据，效率提高了 9.9%。所以，分区表推荐用在基于分区键扫描、非跨区扫描且分区相对均匀的场景。

13.5.2　索引

索引是一种非常有效的加快检索的辅助功能。

像《新华字典》里的索引一样，数据库中的索引都包含两部分信息，一部分是检索键的信息，另一部分是地址信息。检索键的信息是对检索内容的描述，而地址信息是对最终存储数据位置的物理地址的描述。

1. B-TREE 索引

B-TREE 索引是一种在关系型数据库中常用的索引，从“上”到“下”构造了一棵树的结构。树的“叶子”节点就是真实的数据块，树的“枝”节点用来记录数据中建立索引的字段的取值范围（如图 13-9 所示）。查找过程中，在 SQL 查询语句中用 where 谓词对索引字段进行了相应的限制后（例如使用“=”进行限制），就不再进行全表或者全分区范围内的查找了，而是从“根”节点到某一“枝”节点再到某一“叶子”节点逐步深入地查找，同时避开那些根本没有必要扫描的数据块或者索引块，从而降低 I/O 成本。请注意，它的本质是使用二分查找算法，在整个数据集熵最大的情况下，工作效率的提升程度最高。

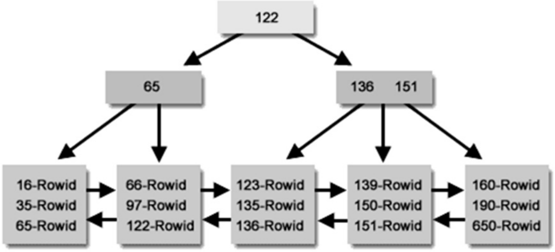


图 13-9 B-TREE 索引

索引在带来便利的同时，也需要付出相应的代价。一方面，索引属于信息冗余性描述，所以肯定会占用相应的磁盘空间。按照索引构建的规则来看，应该会占用 2 倍的索引键空间。另一方面，为了起到加速检索的作用，索引必须与表数据保持一致，也就是说，表中每加入一条数据，索引就需要重新进行一次计算和调整以保持内容信息、地址指针信息与数据表一致，而这些操作也需要消耗 I/O 和 CPU 资源。

2. BITMAP 索引

在索引键的内容高度重复的情况下，例如索引内容是“男、女”、“东、西、南、北”、“S、M、L、XL、XXL”等枚举值，B-TREE 索引通常无法起到加速作用。我们知道，B-TREE 索引的基本思想是数据结构中的二分查找思想，而对于这种高度重复的数据是无法通过这种方式查找且每次都避开剩余数据中的一半数据的。那么，应该如何加速呢？“Bitmap 索引”（位图索引）可以提供相应的功能。

如图 13-10 所示，以英文中记录“东、西、南、北”的索引值“East、West、South、North”为例，在建立索引时要以图中所示的方式进行标注，这样就没有 B-TREE 树来做二分查找了。纵列是 4 个键值，横列是各行的行号或物理地址，如果某行上有这个值就用 1 来表示，如果没有就用 0 来表示，这样就形成了一个二维数组。一旦在用于查询的 SQL 语句中使用 where 谓词对 Bitmap 索引建立的列进行限制，就可以立刻过滤出所有带有这个键值的数据块的物理地址列表。

Region	Row 1	Row 2	Row 3	Row 4	Row 5
North	0	1	0	0	1
South	0	0	1	0	0
East	1	0	0	1	0
West	0	0	0	0	0

图 13-10 Bitmap 索引

因为通过索引键值（也就是枚举值）可以直接指向一个表示行信息的地址数组，所以这种方式的扫描效率很高。以 Oracle 的 RowID 为例，使用 18 字节作为数据行的物理地址，理论上一个 8KB 的块可以存放约 455 个数据行的地址信息。对于枚举类型来说，至少可以避开其他枚举值所在行的扫描。在字段 A 中有 *n* 个枚举值且分布均匀的情况下，检索花费的时间会比没有

Bitmap 索引短，占比为 $\frac{n-1}{n}\times 100\%$。

需要注意的是，索引的使用也不是万能的。如果 where 限制谓词不能通过索引的描述特性避开大部分的数据块扫描，那么索引一定会失效。举个例子，在一个 INT 类型的字段 A 上建立 B-TREE 索引，如果使用谓词

```
SELECT ……
WHERE A<>10;
```

当数据库的 SQL 解释器拿到这个谓词之后，由于无法通过 B-TREE 树的二分查找原理避开相应的数据块，因此只能进行穷举查找，这时索引将会失效。

3. QUBE 模型

如果在扫描时，结果集中的数据量在整个扫描数据集中占比非常小，就表示索引的效果非常好，反之，索引速度就会变慢。这是为什么呢？熟悉 Oracle 的朋友应该知道，Oracle 的 SQL 策略优化是基于一个叫作“CBO 模型”（Cost-Based Optimizer）的东西。CBO 模型从中文直译的角度看就是一种基于成本（Cost）的优化器，这个成本是指访问成本。有一个与 CBO 模型近似的成本评估模型叫作“QUBE 模型”（Quick Upper-Bound Estimation）。这个模型很简洁，也便于我们研究和评估。

QUBE 模型有几个重要的公式：

$$LRT = TR \times 10\text{ms} + TS \times 0.01\text{ms}$$

$$LRT = TR \times 10\text{ms} + TS \times 0.01\text{ms} + F \times 0.1\text{ms}$$

其中，“LRT”代表“Local response time”，“TR”代表“Number of random touches”，“TS”代表“Number of sequential touches”，“F”代表“Number of successful FETCHes”。

这个公式的含义是：

响应时间 = 随机读的次数 × 10ms + 顺序读的次数 × 0.01ms + 处理的记录数 × 0.1ms

在这个模型中有一个假设，就是一次随机读取时间大约为 10ms。这 10ms 是怎么估算出来的呢？以 SCSI 硬盘为例，转速为 10000 ~ 15000rpm，则每秒转动 167 ~ 250 圈，平均每圈耗时 4 ~ 6ms。除了转动消耗的 4 ~ 6ms 外，还有排队时间、传输时间。计算可知，共计约 10ms 完成一次完整的访问。每次每条记录的读取耗时 0.01ms，这个假设的前提是磁盘带宽为 40MB/s，一个数据块（4KB）上平均存放 10 条记录。每条记录的传输时间是

$$\frac{1}{\frac{40\text{MB/s}}{4\text{KB}}\times 10} = 0.01\text{ms}$$

因此，有如下推论。

假设索引的选择率（Selectivity）是 s，表中的记录数是 N，按照 QUBE 模型会得到

$$\begin{aligned}\text{索引访问时间} &= (1+s\times N)\times 10 + s\times N\times 0.01 + s\times N\times 0.1\\ &\approx s\times N\times 10 + s\times N\times 0.1\\ \text{全表扫描时间} &= 1\times 10 + N\times 0.01 + s\times N\times 0.1\\ &\approx N\times 0.01 + s\times N\times 0.1\end{aligned}$$

单位都是 ms。

经过计算，当 $s < 0.001$ 时，使用索引的时间才会小于全表扫描的时间（因为索引访问本身也有成本）。在工程实践中，结合使用缓存等诸多因素，s 通常会取到 10%～20%。如果 s 再大一些，就建议使用全表扫描。在这个讨论中有特殊的数值假设，在实际应用中会使用指标不同的硬件设备，所以会对 s 临界值的估算会有一定影响。在这里，我们只讨论考虑问题的思路，具体的场景要在具体的环境中测算和推定，切莫刻舟求剑。

13.6 稳定最重要

在存储这个层面来说，稳定是非常关键的。这里说的稳定不是指物理层面的硬件不要损坏或者发生其他故障，而是指读写性能方面的稳恒。

不论支持任何种业务，通常我们都希望完成单项任务的时间是以时间 μ 为平均值，以尽可能小的值 σ 为方差的正态分布（如图 13-11 和图 13-12 所示）。

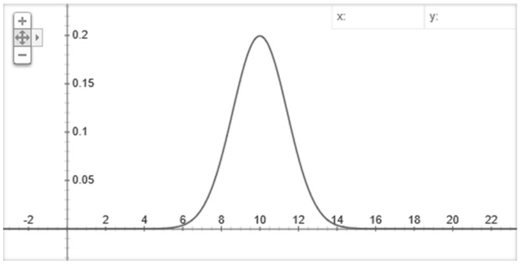


图 13-11　$\mu=10, \sigma=2$ 的正态分布

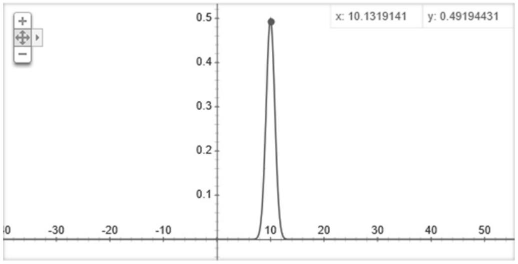


图 13-12　$\mu=10, \sigma=0.8$ 的正态分布

一次标准读写的完成时间，第二种情况通常要比第一种情况理想。我们都有这样的体会：在一次任务中，如果每次完成的时间都稳定在一个值附近，体验就会显得非常平稳。这种稳定的时间长度由于预期性很好，所以可以基于它去做各种相关任务时间的估算——这些时间的估算值也是稳定且可靠的。例如，使用 B-TREE 树索引对表中的非重字段查找进行加速时，时间复杂度为 $O(\log n)$，从“根”到任何一片“叶子”所花费的时间非常稳定。但是，如果不使用这种方式（例如直接使用顺序查找），麻烦就大了。

在不使用索引进行顺序查找的情况下，在对非重字段进行查找时，由于任何一条数据成为被选中数据的概率相同，所以查找的时间复杂度就是 $O(n)$，平均每次理论查找次数为 $\frac{(n+1)}{2}$。然而，每个样本被查找时所花费的时间迥异，第 1 个样本被查找到的时间为 1，第 2 个为 2，第 n 个为 n。查找时间统计模型方差为

$$\sum_{i=1}^{n} i^2 + \frac{(n+1)^2 n^2}{2} + \frac{n^3 (n+1)^2}{4}$$

随着 n 的增大，方差值就会增大。我们看到的现象是：每次查找的时间都和平均值相差很大，查找时间平均值对描述查找时间而言失去了意义——这有悖于统计学对中平均值的定义。

这种不稳定的现象简直糟透了。在一个系统中，如果接连发生这种不稳定现象的叠加，则会导致一系列操作时间的不可预知及不可控，最终导致系统层面的调度成为不可解的问题。存储方面，在硬件环境已经确定的情况下，如果希望写入和读取的速度相对稳定，则需要考虑在统计层面研究读写性能，分散 I/O 热点，并尽量保证对任意数据的读写性能不发生歪斜[111]。

13.7　安全性和副本

13.7.1　RAID

在数据存储环节，最后要考虑的就是安全性问题。这里的安全性不是指防范黑客攻击，而是指数据本身和介质的安全问题。

由于磁盘本身有一定的失效概率，所以需要通过在磁盘上增加数据的副本来保证数据的安全性。在计算机磁盘底层还有一个环节可以调整，那就是磁盘阵列（Redundant Arrays of Independent Disks，RAID，如图 13-13 所示）。

RAID 在服务器上的应用很普及。借助 RAID 卡，硬盘有很多种 RAID 模式可以设置。

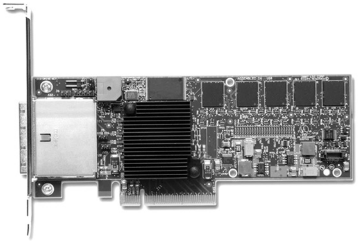


图 13-13　RAID 卡

1. RAID0：条带化（Striping）

RAID0 模式将数据读写分散在两块磁盘上，奇数标号的块在一侧，偶数标号的块在另一侧，写入时轮流对两块磁盘进行写入（如图 13-14 所示）。由于 RAID 卡的带宽通常比磁盘的 I/O 能力高得多，所以这种模式可以让写入速度成倍增长，且与磁盘的数量成正比。读出的时候，由于连续读或者随机读的期望都基于读取每块磁盘上数据的概率均等，所以读出的速度理论上也

111　Skew（Skewness）。

和磁盘的数量成正比。读写的速率上限就是主机的总线（BUS）带宽。

实测时，读写速度通常要比理论值低一些，但档次较高的 RAID 卡速度仍然很快。RAID0 模式可以加快磁盘读写速度。

2. RAID1：镜像化（Mirroring）

RAID1 模式在将数据写入磁盘时生成一个副本，让两块磁盘对应的数据块保存相同的信息（如图 13–15 所示）。虽然这种模式无法对数据的读写进行加速，但是在一块磁盘损坏的时候，另一块磁盘仍然能够正常工作。当换下损坏的磁盘，换上新的空白磁盘时，RAID 卡会自动进行数据同步，再次使两块磁盘保存的信息一致。RAID1 模式可以提高数据的可靠性。

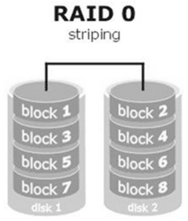


图 13–14　RAID0 模式

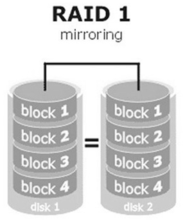


图 13–15　RAID1 模式

3. RAID5：奇偶校验

RAID5 是 RAID0 和 RAID1 的一种折中方案。

RAID5 磁盘阵列至少需要 3 块磁盘（也可以有更多的磁盘），其中 1 块磁盘用来存储奇偶校验数据。如图 13–16 所示，在 Disk0 上写入 A1 块，在 Disk1 上写入 A2 块，在 Disk2 上写入 A3 块，这时写入下一块之前会在 Disk3 上写入一个由 A1、A2、A3 这 3 个块做 XOR 计算而得到的 A*p* 块。

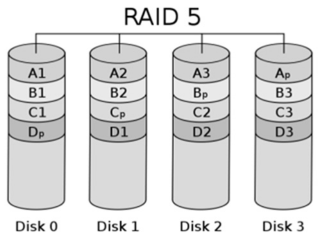


图 13–16　RAID5 模式

$$Ap = XOR(A1, A2, A3)$$

下一轮为

$$Bp = XOR(B1, B2, B3)$$

以此类推。校验块会在整个磁盘组中按照 Disk0→Disk1→Disk2→Disk3 或者 Disk3→Disk1→Disk1→Disk0 的顺序轮转。

一旦一块磁盘发生损坏，根据 XOR 计算的原理：

$$A1 = XOR(Ap, A2, A3)$$

$$A2 = XOR(A1, Ap, A3)$$

$$A3 = XOR(A1, A2, Ap)$$

只要不是一次损坏 2 块以上的磁盘，数据都可以无损恢复。

4. RAID10：条带化 + 镜像化

RAID10 模式相当于每 2 块磁盘先做 RAID1，然后在组合的 RAID1 磁盘组上再做一次 RAID0（如图 13–17 所示）。这种模式兼有条带化和镜像化的特点，在生产环境中应用最为广泛。

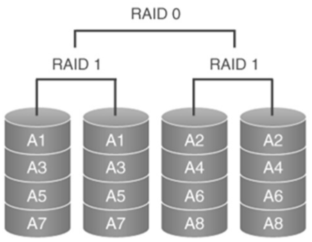


图 13–17　RAID10 模式

RAID0、RAID1、RAID5、RAID10 的磁盘使用率、可靠性等工作指标如表 13–1 所示。假设每块磁盘的可靠性为 P_{disk}，RAID 卡的可靠性为 P_{raid}，每块磁盘的读写速度为 V，用 n 块磁盘组成阵列（为了讨论方便，n 取大于 2 的偶数，并假设 RAID 卡读写性能的上限为无穷大）。

表 13–1　磁盘工作指标

	可 靠 性	读写速度	磁盘使用率
RAID0	$P_{raid} \cdot P_{disk}{}^{n}$	$n \cdot V$	100%
RAID1	$P_{raid} \cdot (1-(1-P_{disk})^{n})$	V	$\frac{1}{n} \cdot 100\%$
RAID5	$P_{raid} \cdot P_{disk}{}^{n-2}$	$(n-1) \cdot V$	$\frac{n-1}{n} \cdot 100\%$
RAID10	$P_{raid} \cdot (1-\frac{n}{2} \cdot (1-P_{disk})^{2})$	$\frac{n}{2} \cdot V$	50%

从这些值上来看，通常我们都会认为 RAID10 除了使用率不理想以外，其他方面的表现都还不错。一般情况下，可能除了能从磁盘厂商处拿到 V 的数据以外，P_{disk} 和 P_{raid} 这类数据是拿不到的。所以，只要注意在同一个 RAID 组里的磁盘尽量不要选择同一批次的就好，以免由于同一批次生产的磁盘失效时间相近导致整体失效概率升高。

13.7.2　软冗余

除了 RAID 这种在单机上做冗余的方案以外，现在业界使用比较多且性价比比较高的存储介质是 HDFS 或者 Ceph。这两种方案都是基于网络的分布式存储方案，而且兼有读写分散和冗余的功能。

HDFS 默认在配置文件 hdfs–site.xml 中有 dfs.replication 配置项，默认值是 3，也就是将一个 HDFS 数据块在整个网络节点中保存 3 份。这个选项也可以修改，通常以单台主机的有效性 P_{host} 来计算，在 NameNode 充分高可用的情况下，数据的可靠性就是 $1-(1-P_{host})^{3}$。这种设置一般被认为是在生产环境中性价比比较高的方案。如果每台主机的磁盘都已经做过 RAID10，也可以适当降低 dfs.replication 配置项的值。

HDFS 的写入速度也是比较快的。由于和数据库相比，HDFS 单纯面向数据文件进行读写操作，所以其中没有锁存在。而且在大量的写操作中，NameNode 会将数据写分散到每台 DateNode 中去。理论上，在由单台读写速度为 V 的主机组成的 n 个节点 3 冗余环境中，写速度能够达到 $\frac{1}{3}\cdot nV$；读速通常就无法提升了，因为这涉及 Client 端的瓶颈问题。

13.8 本章小结

这一章主要讨论了数据存储时需要注意的可靠性和性能问题，以及存储介质是否便于未来统计工作的开展。这些讨论对业务中的存储选型会有一定的提示作用。

Oracle 数据库被认为是计算机商用发展过程中存储技术的集大成者。对 Oracle 数据库产品存储特性的研究，对做好存储架构设计有着非常重要的指导作用，即使是在对事务不敏感的场景中也有极高的借鉴价值。除此之外，互联网公司在面对大型的数据存储需求时，通常会考虑使用 HDFS with LZO 压缩的方案。因为在存储环境中，大多数时候 CPU 是比较空闲的，而磁盘上存储的信息中大部分是文本信息，所以压缩率[112]很高，在一些情况下几乎可以达到 20%。这种方案还能带来一个好处，就是节省宝贵的磁盘 I/O 资源。这种用富余的 CPU 资源换取存储空间的方案是不错的选择。

无独有偶，这种思路在一些提供云服务的公司还有其他玩法。在很多社交软件中都有这样的用例：先将用户上传的高清照片裁剪成 512×512 像素的正方形大尺寸头像，再调整（Resize）成 128×128、64×64、32×32 等尺寸。为了充分利用 CPU 资源，节省存储空间，服务器上只存储 512×512 像素的大尺寸头像，每次请求头像时调用这个服务商提供的 URL，参数为大尺寸头像的 ID 和请求尺寸，实时进行 Resize 操作，延迟很低。

本章涉及的所有查找性能指标都是近似理论值，在实际的生产环境或测试环境中很难观察到这样的测量数据。由于在实际环境中的 CPU 指令流和 I/O 吞吐的随机性很高，因此在数据存储环节的各种性能观测都混杂着一定的未知干扰。除此之外，不同的系统软件（尤其是那些非开源的软件）及其不同版本对这些存储和查找的算法实现也有不同程度的优化。这样一来，观测值就与理论值有了差异。在真实环境中进行测试时，还是建议用真实场景中的读写特性作为测试样例进行测试，以逼近实际生产环境的性能。

112 压缩率（Compresson Ratio）由压缩后占用空间大小除以压缩前占用空间大小得出。

第 14 章　数据统计

将数据收集到数据库或者数据仓库中以后，就可以进行统计和分析了。

统计的目的就是将大量离散的数据经过统计计算，形成集中性的指标或者粗粒度单位。统计是在海量数据分析之前必须要做的一步工作。从本身的目的来说，统计将大量复杂的数据个体宏观特征化，简化了数据解读者对认知对象的了解过程。换句话说，数据统计的意义主要在于数据画像，节约阅读时间。

14.1　此“统计”恐非彼“统计”

说到“统计”，很容易让我们想到在学校里学习的“统计学”课程内容，只是因为统计任务已经被规约成任务单一的指标计算或同比和环比计算，所以我们在日常运营工作中用到的统计知识相对比较简单。而且，在全样本统计成本越来越低的情况下，通常不会用到太多高级统计学中有关从抽样误差反推到全样本误差的知识。不过，这仍然是统计，所以我们在统计学中需要注意的问题，在这里也一样需要注意。

既然数据统计的意义主要在于用数据为认知对象画像，那么数量统计、数值加和、数值平均、方差、同比、环比等就成了绝大多数单位在运营过程中所需的基本数据资源。在这样的情况下，我们就把这种一般性的“操作”当成“统计”来介绍，把“高级技巧”放在后面作为分析的内容来讲解。

据我所知，现在很多人都有这样的误解：在数据库里使用 AVG、SUM、COUNT 这样的语句进行统计是一件档次很低的事情，因为操作太简单，有一点 SQL 语言基础的人都能做到。但我并不这么看。统计通常用于把认知对象的维度从低变高，从高变低，从简易变得复合，从复合变得简易。这种认知维度升降中的变化，可以帮助认知对象在观测过程中处于复杂度适中的状态，从而为人们进行认知与归纳总结提供便利。

我们先说降维认知。当数据处于高维状态的时候，我们一般喜欢把它们降维到适当的状态来认知。数据挖掘领域使用的“数据立方体”（Data Cube，如图 14-1 所示）就是一种降维认知手段。在描述一个维度极为丰富的高维数据记录组（例如销售记录）的时候，这种记录可能会

包含销售时间、销售地点、销售类目、销售金额、销售数量、折扣率、售货人员等各种维度的信息，而且会有数以百万计的数据记录量。假设有 m 个维度和 n 条销售记录，那么这些数据会形成一个 $m \times n$ 的巨型矩阵。但是，销售额是一个对加和值敏感的数据种类，在使用

```
SELECT SUM(XXX)
GROUP BY XXX
```

之后，就会忽略一些我们不那么关心的维度数据，并将一个确定的加和值返回，从而代替数以万计的具体销售记录。这种就是典型的降维认知——只关注最为关键的数据维度，只关注一个统计值而非单个个体值。

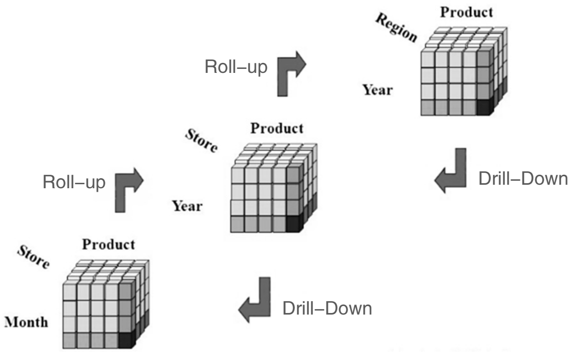


图 14–1 数据立方体

除了降维认知以外，升维认知（如图 14–2 所示）的使用也很广泛。

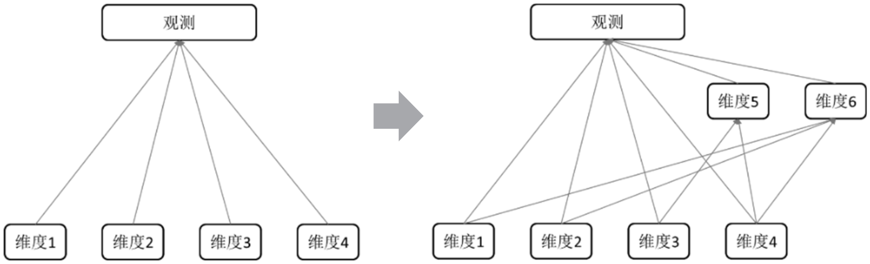


图 14–2 升维认知

在表示股价或期货价格的 K 线图（如图 14–3 所示）中，横坐标是时间，K 线图上的每一个柱形都表示 4 个价格信息，分别是开盘价、高价、低价、收盘价。这种数据蕴含的信息看上去很贫乏，因为只有时间和价格，对市场多空情绪的反映有些不直观。所以，股票分析家们又发明了很多技术指标，这些指标无一例外，都是用 K 线图上的数据进行计算所得到的升维指标数据，MA（Moving Average，移动平均线）的计算就是其中之一。

图 14-3　股票价格 K 线图

在一个 MySQL 数据表 Price 中有字段 CLOSE 作为收盘价的字段（如表 14-1 所示）。

表 14-1　Price 表信息

TIME	CLOSE
2016/10/3	104.15
2016/10/4	102.33
2016/10/5	102.25
2016/10/6	101.54
2016/10/7	101.86
2016/10/10	102.43
2016/10/11	102.90
……	……

计算收盘价 *MA*，应该使用下面的步骤。

$$MA_n(period) = \frac{\sum_{i=0}^{period-1} Price_{(n-1)}}{period}$$

以 3 天移动平均线指标为例，先建立一个以 ROW_NUM 为升序伪列的表。

```
SET @ROWNUM=1;

CREATE TABLE TEMP AS
SELECT @ROWNUM:=@ROWNUM+1 AS ROW_NUM,TIME,CLOSE
FROM PRICE
ORDER BY TIME;
```

建立一个指标表放置 MA 数据。

```
CREATE TABLE EMA AS
SELECT
  R3.ROW_NUM, R3.TIME, (R1.CLOSE+R2.CLOSE+R3.CLOSE)/3 AS MA
FROM
(
  SELECT ROW_NUM, TIME, CLOSE
  FROM
  TEMP
) AS R1
LEFT JOIN
(
  SELECT ROW_NUM, TIME, CLOSE
  FROM
  TEMP
) AS R2
ON R1.ROW_NUM=R2.ROW_NUM-1
LEFT JOIN
(
  SELECT ROW_NUM, TIME, CLOSE
  FROM
  TEMP
) AS R3
ON R2.ROW_NUM=R3.ROW_NUM-1;
```

这样，MA 列中就是 3 天移动平均线的值了。

同理可以计算任意多天的移动平均线的值，也可以考虑用游标等技术实现。一些技术流分析人员观察短期移动平均线上穿或下穿长期移动平均线的情况，并以此作为入场信号，因为这体现了近期变化的趋势改变，这种方式要比直接观察 K 线图容易很多。此外，EMA（Exponential Moving Average，指数平滑移动平均线）、MACD（Moving Average Convergence Divergence，指数平滑异同移动平均线）、RSI（Relative Strength Index，相对强弱指标）、BOLL（Bollinger Bands，布林线指标）等指标也是通过对 K 线图上价格数据的统计计算得到的，对分析价格数据有很大的辅助作用。这些引入的指标给人们提供了更多关于市场价格的认知维度，虽然它与原始的 K 线图价格维度不是正交的，但同样带来了信息。分析师们会通过这些新加入的维度，以及已有 K 线图上开盘价、高价、低价、收盘价的数值，进行有关是否到达支撑位置、是否形成有效突破等市场情况的判断。现在你不会觉得这些简单的 SQL 语句没有档次了吧？

在结合信息论的相关理论后，一些读者可能会有疑问：这些信息量都已经包含在原始数据中了，这种升维操作应该是一种没有信息量的数据信息，那么这种升维还有意义吗？这个问题可以这样理解。首先，信息论研究的问题仅仅是数据中蕴含的信息量的问题，讨论的是围绕信息量的传输、压缩等一系列“失真”或“非失真”程度的问题，这是它的研究边界。而我们刚

刚讲到的，维度中蕴含的信息即使已经处于信息量的下界，也只是不能被更短的数据来表示而已，并不意味着这些信息对人类进行认知而言已经足够“好”了。想想看，这些数据信息中至少不包括升维或降维算法的信息，如果要说信息量的话，这部分同样应该归入整个讨论的边界。在升维后体现出来的新的或者原来不直观的特性，很可能帮助我们找到一些新的认知线索。这就是数据统计的意义。

升维也好，降维也罢，都是把维度调整到的合适的程度来归纳和认知，具体的理论依据会在第 16 章介绍。

14.2　要精确还是要简洁

统计作为一种升降维的手段，已经成为非常重要的认知工具。

表示内能大小的“温度”是最平民化的统计指标。我们都知道，物体的温度表示的是物体内部分子或原子的无规则热运动动能。对大量无规则运动的原子或分子进行观测，或者对每个原子或分子的运动速度进行描述，都是不太现实的。最终，还是在宏观层面使用了“温度”这样一种方式来描述，才解决了问题。虽然“温度”和“内能”确实不能直接画等号，也确实存在内能增加而温度不增加或者温度增加而内能不增加的情况，但是温度这一指标的确完成了一种从微观大量数据到单个宏观描述数据的映射过程。

我们在量化研究一个对象时可能会面临这样一种困扰：不知道怎么研究才更合适。像刚才说的，使用温度来描述冷热程度，而最终描述的是宏观原子或分子的平均动能大小，这种方式适合研究对象是宏观的情况。当研究微观层面的对象时，就只能针对更小的单位去描述了，后果就是描述的对象数量会增加。

“过犹不及”[113]这种中庸的思想在统计认知领域是一种无形的准则。由人类自身认知能力局限性所形成的认知导向是每个人（尤其是数据分析人员）必须时刻牢记的。所谓认知能力局限性主要是指人类自身认知客观事物时用较低的成本所能掌握的维度。下面我们通过几个例子好好体会一下这种感觉。

人类长成什么样子？在向他人叙述人类的外形特点时，我们会使用怎样的语言？

“人有一个头颅，有躯干，有双臂双手，双腿双脚。头颅上有头发，有一双眼睛，一双耳朵，一个鼻子，一张嘴。直立行走。”这种描述就已经很好了。可是，如果换一种方式，就显得比较奇怪：“人由 206 块骨头和 600 多块肌肉组成，有 500 多万亿个细胞，有 10 万根左右的头发，还有 28 到 32 颗牙齿……”乍一看，第二种方式好像数字相对精确，而且细节比较多，可

113　出自《论语・先进》：“子贡问：‘师与商也孰贤？’子曰：‘师也过，商也不及。’曰：‘然则师愈与？’子曰：‘过犹不及。’”译为：孔子的学生子贡问孔子，他的同学子张和子夏哪个更贤明一些。孔子说，子张常常超过周礼的要求，子夏则常常达不到周礼的要求。子贡又问，子张能超过是不是好一些。孔子回答说，超过和达不到的效果是一样的。

是这些数字对区分人类和其他生物的外形帮助不大，所以我们平时对它们不那么关心。难道我们有机会拿这些数字来甄别一个人和其他生物的区别吗？——大多数人又不是法医。因此，在这种情况下，精确就显得没有太大的必要了。

我们需要在“精确”和“简洁”这对矛盾之间找平衡。越“精确”的东西描述起来越繁杂，因素维度越多；越“简洁”的东西描述起来越简单，因素维度越少。通过升维或降维将数据维度调整到比较合适的数量和规模进行研究才比较有效。我的体会是：在一个公式型的关系描述中，涉及的对象最好只有 3 ~ 5 个甚至更少；如果是由多个关系形成的复杂关系，可以把它们化解成多组由 3 ~ 5 个对象形成的关系。这种维度之间的关系相对更容易认知和讨论，尤其是在做对照测试的时候。

14.3 统计是万能的吗

作为认知工具，统计必然是对认知有帮助的，但它同样有自身的局限性。

从信息表述的角度来说，统计的指标信息都是总括性的信息，无论方法如何科学，都会丧失一部分原始信息的内容。所以，从这个角度来看，统计信息更像是一种有损压缩，这就是它的局限性所在。

传统的加和值、平均值、最大值、最小值、样本量值等指标的使用极为普遍，在几乎所有的场景中都可以使用并尝试解释它们之间的关系。现在的社会学、宏观经济学、微观经济学研究会更多尝试这些值的复杂组合。

在社会学研究中有一个叫作“基尼系数”的研究指标。

基尼系数是 1943 年美国经济学家阿尔伯特・赫希曼[114]根据洛伦兹曲线[115]（如图 14-4 所示）定义的判断收入分配公平程度的指标。基尼系数是一个比例数值，取值范围在 0 和 1 之间，是国际上用来综合考察居民内部收入分配差异状况的一个重要的分析指标。

洛伦兹曲线用于比较和分析一个国家在不同时代或者不同国家在同一时代的收入分配情况。作为一种总结收入和财富分配信息的便利的图形方法，洛伦兹曲线得到了广泛应用。

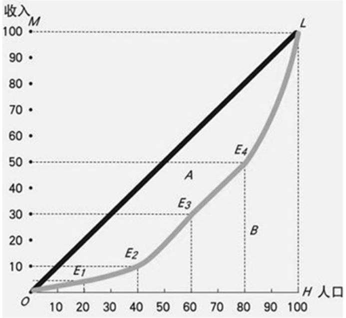


图 14-4 洛伦兹曲线

114 阿尔伯特・奥图・赫希曼（Albert Otto Hirschman，1915 年 4 月 ~ 2012 年 12 月），知名经济学家、思想家，著有《叛离、抗议与忠诚》及《反动的修辞》两本政治学经典著作。

115 在一个总体（国家、地区）内，以“最贫穷的人口计算起一直到最富有人口”的人口百分比对应其他各个人口百分比的收入百分比的点组成的曲线。为了研究国民收入在国民之间的分配问题，美国统计学家 M.O.洛伦兹（Max Otto Lorenz，1876 年 9 月 ~ 1959 年 7 月）于 1907 年提出了著名的洛伦兹曲线。

通过洛伦兹曲线，我们可以直观地看到一个国家收入分配平等或不平等的状况。画一个矩形：矩形的高是用来衡量社会财富的百分比，将其分为 5 等份，每一等份表示 20% 的社会总财富；在矩形的长上，将家庭从最贫者到最富者自左向右排列，也分为 5 等份，第 1 个等份代表收入最低的 20% 的家庭。在这个矩形中，将每一等份的家庭所有拥有的财富的百分比累计起来，并将相应的点画在图中，就得到了一条曲线，这就是洛伦兹曲线。整个洛伦兹曲线的坐标系是一个正方形，正方形的底边（即横轴）代表收入获得者在总人口中的百分比，正方形的左边（即纵轴）显示各个百分比人口所获得的收入的百分比。从坐标原点到正方形相应另一个顶点的对角线为均等线（即收入分配绝对平等线），这种情况一般不存在。实际收入分配曲线（即洛伦兹曲线）都在均等线的右下方。

赫希曼根据洛伦茨曲线提出的判断分配平等程度的指标，设实际收入分配曲线和收入分配绝对平等曲线之间的面积为 A，实际收入分配曲线右下方的面积为 B，并用 $\frac{A}{A+B}$ 表示不平等程度。这个数值称为“基尼系数”或“洛伦茨系数”。如果 A 为 0，则基尼系数为 0，表示收入分配完全平等；如果 B 为 0，则基尼系数为 1，表示收入分配绝对不平等。收入分配越趋向平等，洛伦茨曲线的弧度就越小，基尼系数也就越小；反之，收入分配越趋向不平等，洛伦茨曲线的弧度就越大，基尼系数也就越大。

基尼系数的实际数值只能在 0 到 1 之间。基尼系数越小，说明收入分配越平均；基尼系数越大，说明收入分配越不平均。国际上通常把 0.4 作为贫富差距的警戒线，大于这一数值就容易出现社会动荡。按照联合国有关组织的规定，基尼系数反映的收入分配情况如表 14–2 所示。

表 14–2　基尼系数反映的收入分配情况

取值界限	含　　义
小于 0.2	收入绝对平均
0.2 ~ 0.3	收入比较平均
0.3 ~ 0.4	收入相对合理
0.4 ~ 0.5	收入差距较大
大于 0.5	收入差距悬殊

前面提到，基尼指数通常把 0.4 作为收入分配差距的“警戒线”。根据黄金分割律，其准确值应为 0.382。一般发达国家的基尼系数在 0.24 到 0.36 之间，美国偏高，为 0.45。根据国家统计局公布的数据，中国的基尼系数 2015 年为 0.462，2014 年为 0.469，2013 年为 0.473，2012 年为 0.474，2010 年为 0.481。

基尼系数的计算其实已经属于经济指标里计算方法比较复杂的了。这些指标的制定大都没有严格的推导依据，只要经过观察、归纳，然后建立一系列的量化解释就够了。曾有不少学者

诟病基尼系数的局限性。例如，因为没有严格的计算标准，所以“收入”界定方式的差别使不同的组织在计算时使用了不同的统计口径。再如，无法解释基尼系数较大的国家社会仍然稳定，基尼系数较小的国家社会却不稳定的例外现象。此外，社会范围的圈定也是一个非常有争议的环节：是以市、省（州）、国家（地区）圈定，还是以更大的范围圈定？这个范围怎么设置更合理抑或无所谓？……

在这里谈谈我的观点。首先，无论怎样设计统计指标来做辅助性的认知，都是可以尝试的方案，因为试探性的解释本身就是认知的必经过程，这一点几千年来从未改变。其次，由于统计方法本身是一种有损压缩的升降维方式，所以对引入的误差或“变形”的影响在未来参与计算时是否会产生不良后果的问题需要谨慎处理。最后，在统计过程中，如何继续通过建模来尝试解释指标之间的关系，其具体方法将在第 15 章讨论。

14.4　注意性能

在统计性能方面，尽量使用可以进行分布式计算的方式或者模型，以充分利用服务器各方面的资源。以 Hive 为例，在一次大量数据的统计过程中，如果覆盖的块均匀分布在每个节点上，那么在进行统计计算时就不会出现负载歪斜（一个集群中只有部分主机的资源非常繁忙，其他主机都处于空闲状态，这通常是由架构规划不良导致的）的情况。

此外，就是避免在全局使用类似 DISTINCT 这种不满足分配律的操作。这种操作的含义是在全局范围内做去重操作，也就意味着所有的数据必须拿到单台主机上做去重操作，而这会使一台主机成为整个计算的瓶颈。

在分布式计算领域用到的负载分散技巧主要是计算资源的哈希问题。在以 Key-Value 为计算基础的 Map-Reduce 分布式计算模型中，通常要把 Key 设计得更为合理，让它能够均匀分散在多个节点上，从而提高计算效率。

14.5　本章小结

本章讨论了有关数据统计的问题。这里说的统计泛指所有通过数据的统合计算产生的抽象性指标系统。这种认知方式会帮助观察者从大量纷杂的个体对象中归纳出某种宏观性的表象信息。

在任何与从宏观角度认知大量事物特性有关的研究领域，例如社会学、经济学、生物学，以及更为底层的统计物理学等方面，科研工作者也都是通过对事物宏观特性的统计归纳来试探性地进行解释和分析的，这个过程带有明显的“试错”意味。数据科学在其中所起的作用，就是帮助人们用辩证的、量化的方法过滤那些不可靠的数据假说。在数据统计这个环节中的归纳，就是对不同数据对象进行升降维的尝试而已。

第 15 章　数据建模

数据建模是一项非常重要的技能，尤其是对数据分析师来说，就像吃饭、睡觉、走路甚至呼吸一样不可或缺。

数据建模，也可以叫作“数学建模”（这两个概念研究的问题几乎没有差别）。关于数学建模的例题和方法论等，现在市面上的很多资料中都有讲述，方法各异，形形色色。如果想从纷杂的方法中总结出一些相对完整的、泛用性好的方法论，我觉得也不难——只要抓住数据建模和数据分析的本质，思路很快就会有了。

数据建模其实是一门为了解决数据之间定量的逻辑关系问题的学科，是一种通过对数据进行分析处理，定量描述其间关系的方法论学科（如图 15–1 所示）。数据建模贯穿在我们接触过的各种学科中，例如物理学、化学、地理学、天文学，这些学科的研究对象也是五花八门、种类繁多，但其本质都是在精确地定量研究这些对象的转化条件、转化规律或逻辑。

数据建模基本可以分为如下 4 个步骤。

（1）观察记录

用相对准确或精确的方式，通过观测来记录观测值数据。

（2）猜测假说

根据观测值做一个逻辑关系“拟合”的假说，假定数据之间的逻辑关系，通常使用方程、函数作为表达形式。此过程中伴随着对残差的分析，并要确保残差足够小（小到大多数人能认可的地步）。

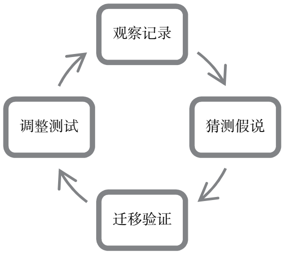


图 15–1　数据建模

（3）迁移验证

数据建模中需要一个证明和分析的过程，也就是把这个逻辑关系换成一组由他人观测的或者在其他环境中观测的数据代入验证，看看误差值是否与预期一致，是否同样足够小（小到大多数人能认可的地步）。

（4）调整测试

如果发现残差比较大，就要检查实验过程中是不是有疏漏，条件约数是不是完整，计算过程是不是缜密。如果有问题，就需要进行相应的修正，例如对约数条件叙述的修正或者对逻辑关系假说的修正。

15.1 模型是宝贵的财富

这里说的“模型”实际上是指一种数据之间定量转化的关系或者关系假说。如果这个“模型”已经经过验证，那么这个假说就是成立的（或者说模型成立）。

模型的获取有如下 3 个渠道。

（1）有经验的从业人员会把他的知识总结为一种经验的陈述。这种经验的陈述如果在定量方面有了比较精确的结论并可以验证，那么这种模型就是一种经验模型。在实际工作中被广泛使用的“二八理论”[116]就是一种典型的经验模型。很多人在使用二八理论进行判断时，已经不会先去做调查了，而是直接假设这种理论在自己工作的场景中成立。

（2）通过大量的数据挖掘或者机器学习算法进行归纳和总结，并且在验证集合上能够通过验证。这里强调一定要“在验证集合上通过验证”，目的是再次强调不要出现过拟合的情况，而且这个环节通常是刚入门的分析人员比较容易出现错误的地方。

（3）向咨询公司购买模型。世界上有很多著名的咨询公司，例如麦肯锡（Mckinsey）、埃森哲（Accenture）、SAP、普华永道（Price Waterhouse Coopers Consulting）等。这些机构每年都在各个行业发布一些数据翔实的调查结果，这些数据可以当成模型使用。但在某些领域，仍然需要数据解读者通过分析和再次建模进行拟合来完善模型。

为什么说模型是宝贵的财富？就是因为模型里凝结了大量的数据统计和分析结果。这些结果的获取耗费了非常多的人力、物力、财力，而且对消除不确定性、降低试错成本有很大的帮助。所以，很多大型企业在做决策时都喜欢购买这类数据模型的成品甚至半成品作为参考。一份数据分析报告的价格从十几万到上千万元不等（主要由分析的粒度与实践指导价值的多寡而定）。

15.2 量化是关键

一个“模型”之所以成为“模型”，是因为其本身拥有描述数据之间关系的能力，关键就在于量化精确。

还记得我们在本书最开始提到的关于天文历法的问题吗？“太阳东升西落”、“四季夏热冬冷”这种程度的描述虽然有助于消除不确定性，而且有一定的正确性，但量化很不到位，所以不仅不能称为科学，就连价值几何也需要画个问号。

116 “二八理论”，也叫“二八定律”，又名“80/20 定律”、“帕列托法则（定律）”（也叫“巴莱特定律”）、“最省力的法则”、“不平衡原则”等，被广泛应用于社会学及企业管理学等。1897 年，意大利经济学者帕累托偶然注意到 19 世纪英国人的财富和收益模式。在调查取样中发现，大部分财富流向少数人手里。同时，他还从早期的资料中发现，其他国家都有这种微妙关系一再出现，而且在数学上呈现出一种稳定的关系。于是，帕累托从大量具体的事实中总结：社会上 20% 的人占有 80% 的社会财富。这种关系后来被泛化到其他商业场景，即 20% 左右的客户对象贡献了 80% 左右的价值。

为什么量化如此关键？量化不精确一定会出现严重的问题吗？我想应该这样考虑。

首先，抛开科学与否的问题不谈，量化是唯一能够用来量化精确程度的前提。本书一直坚持一个观点，那就是观测值才是一种客观认知下的“正确”值，一切假说和推导最终都要落实到与观测值的比对和对误差的衡量上。所以，一个模型准确与否，必须先做量化才能讨论。而且，一个模型是否足够精确，以及另一个模型与此相比是更精确还是更不精确，也都严重依赖量化这一前提。

其次，量化的精确性也同样是提高工作效率、降低成本的重要因素。对一件工业品来说，每种制造方案的成本精确量化会直接把成本和收益体现在盈利模型中。

最后，没有量化的内容就无法计算，无法计算就不能描述差距，不能描述差距就无法针对方法的改进和提高作出评价。没有量化就没有进步的标尺——这个结论的毁灭性太强了。

所以，如果要进行数据分析和研究，想得到相对精确的认知，就要先做准确的量化，哪怕这种量化看上去还不够完美。

15.3　该算法出马了

任何算法都是为模型服务的，也就是为最终解释或换算量化值的关系服务的，所以在建模环节一定会用到相当数量的算法。对非计算机专业人员来说，虽然算法显得很神秘，难度也很高，但我们只要明白算法的本质是“通过计算机存取单元的访问调度和数值计算来实现模型”就足够了。

在第 11 章“算法学”中，我们已经讨论了在计算机领域使用算法的关键思路。那么，数据建模中使用的算法和算法学中提到的算法有什么异同？答案是：没有异同。数据建模环节使用的算法，大都带有探索性数据分析的意味，故以关联性算法、概率统计算法、线性/非线性模型拟合算法等为主。因此，机器学习和数据挖掘领域的任何一种算法都可以作为建模的手段。

请注意，在任何情况下我们都不希望把一个模型叙述得过于复杂。除了因为对简洁性的追求是一种天然的、描述成本最低的原则以外，还有一个非常重要的原因——模型设计越复杂，同样越容易发生过拟合（也就是失去泛化性，在应用于其他场景时频繁失效）。

计算机出现之前，在使用笔算对模型进行推导的过程中，最典型、最简单的场景就是用最小二乘法对残差求极值。由于笔算时便于使用解析解的形式来表达解的过程，所以这种形式非常注重推导过程和解释意义的严谨，这对一些在笔算层面推导复杂的过程来说尤为艰难。不过，这种情况在计算机出现之后好了很多，使用迭代法进行优化的建模过程变得非常方便，甚至很多框架提供的类只需要几行代码就能实现。这样，问题就剩下在统计过程中计算某些值的数量，观察这些数量之间的关系，以及建立这些数量之间的假说了，其他问题都可以通过计算机的加、减、乘、除运算得到结果，或者通过使用迭代法找到损失函数的极值来完成。所有求

偏导的过程再也不需要用笔算来解决了，直接利用定义中的

$$\lim_{\Delta \to 0} \frac{f(x+\Delta)-f(x)}{\Delta}$$

取 Δ 为一个非常小的浮点数即可。一切在数学理论上连续可导的损失函数都可以用这个方法来解决，这也是用计算机做残差优化的基本算法保障。

15.3.1 统计学模型

大部分传统的机器学习内容都是统计学模型，包括我们前面说的线性关系也可以放在这个范畴。有些模型关系不是典型的通过求 *Loss* 函数的极值得到的，例如朴素贝叶斯

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

就是纯粹通过统计得到的模型关系。还有隐马尔可夫模型（在 11.8.1 节提到过），这种学习过程的结果是得到一个概率转化矩阵。

使用统计学模型得到的内容，大都是对事物之间概率的关系解释，对普通 SQL 编程掌握较好的工程人员都能很好地理解这一点。在这种模型的归纳中只需注意：是否可以通过 AB 对照的测试方法来验证自己的观点？如果不能验证，那么这种模型仍然是一种过拟合模型。即使不使用 AB 对照，也要使用类似的方法进行验证，例如另外挑选与期望结果相似的验证样本进行统计（就像在机器学习中使用的验证集）。采取这样的方法都是为了避免在个案上归纳出的特例被误泛化到其他场景中。对于维度越多、因素越复杂的场景，这种工作的必要性就越大。

15.3.2 线性关系

线性关系是所有关系中最简单的一种模型，也因此成为许多人希望一开始就尝试利用的关系假说。这个关系假说就是一个多维向量 x 和 w 矩阵的内积关系与偏置 b 的加和的形式。

$$f(x)=wx+b$$

这个形式在本书中多次提到，我们对它已经很熟悉了。尤其在 x 是一维向量的时候，公式会退化成 $f(x)=wx+b$ 的形式。在训练过程中，要注意观察损失函数 *Loss* 的下降情况，确认 *Loss* 已经下降到业务允许的范围内才可以使用。在使用最小二乘法作为 *Loss* 函数时，*Loss* 函数的值就是样本点和拟合值之差的欧氏距离的平方和。

15.3.3 复杂的非线性关系

太复杂的线性和非线性回归的叠加，产生的回归模型往往会异常复杂，而且给特征的提取带来了很大的挑战。在很多新的研究领域，会把特征提取这个环节也交给计算机来做。用什么来做？用深度学习。

深度学习就是各种深度神经网络的组合应用。在第 11 章，我们已经对 BP、CNN、RNN 这 3 种网络进行了介绍。由于网络的输入层维度可以有几十万甚至上百万维，网络的深度也可以有三四十层甚至更多，所以其中容纳的线性和非线性空间解也是极为丰富的。

在训练过程中，可以通过梯度下降的方式寻找一个最为合适的 W 矩阵

$$W=\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & \cdots & \cdots & w_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

从而在没有产生过拟合的情况下使损失函数极小化。

不过，这里的问题也是显而易见的。即使这个模型最后产生的召回率和精确率很高，得到的 W 矩阵实际上也只一个纯粹的“黑盒子”——没有任何明确物理解释的模型。老实说，这种方法在“科学性”方面可能只是比较好地满足了“精确性”，而在“体系性”上让人觉得略显不足。对深度神经网络的研究，目前还有很多问题没有解决。对于纯粹以应用为导向的环境来说，只要能够提高业务水平，使用这样一个“黑盒子”或许没有人会介意，例如使用深度神经网络来做推荐系统。而反过来，如果希望在这个模型中得到业务层面的指导知识就不行了，因为这种推导不可逆，即使发生了转化率的提高，我们也无法推定是由哪个或哪些因素造成的影响。例如，卷积层上的权重 w 没有统计学加权平均中的权重解释，通过可视化查看卷积层的输出，就是一块一块的光斑，这就导致无法在现实中通过改进构成业务的因素来最终实现整体业务的提高（如图 15-2 和图 15-3 所示）。所以，在这种情况下，如果想要逆向理解影响因素，就需要使用统计学中的 AB 对照方法。

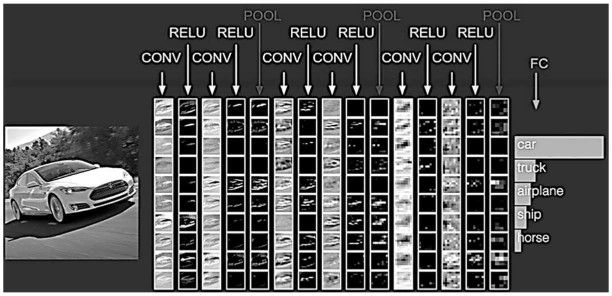


图 15-2　卷积网络分类器

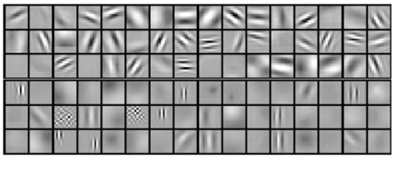


图 15-3　卷积层可视化

15.4　算法的哲学

所有有关数据建模方面的算法，套路都已经固定了（如图 15–4 所示）。

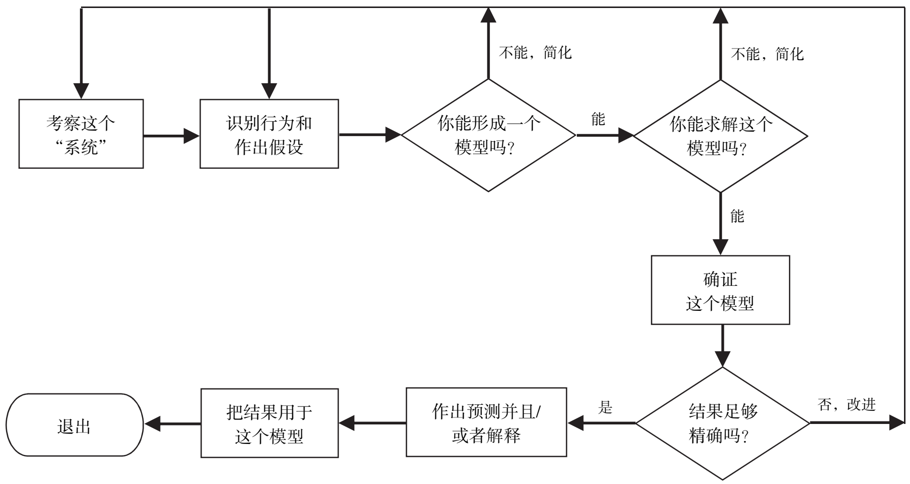


图 15–4　模型的构建流程

第 1 步：拿到一定量的样本数据。

第 2 步：从样本数据中拿出一定量的数据做“训练”，留下一些数据准备做验证。在训练的过程中，把误差（损失函数）*Loss* 描述成一个与待定系数有关的凸函数，或者通过统计与概率获得结果。

第 3 步：对损失函数进行优化，使其逐步收敛到满足模型精度需求的程度。

第 4 步：使用剩余的数据来做验证（Validation），看看在第 3 步训练中得到的模型的召回率和精确度是否仍旧保持不变。如果保持不变，就说明这种模型的泛化性比较好；如果发现召回率和精确度不如训练集的表现好，就说明训练中产生了过拟合现象，需要进行调整。

第 5 步：这是一个可选步骤，叫作测试（Testing）。拿一些训练样本以外的数据对模型再进行一次检验，看看召回率和精确度是不是和训练集的表现一致。如果不一致，则仍需对模型进行检验（Review），以找到问题所在并加以改进。

第 4 步和第 5 步所做的事情类似，在很多实验环境中不会进行第 5 步。而在一些即将商用的模型演进过程中，会通过第 5 步反复测试，直至确认没有问题才会投入商用。

这个“五部曲”就是建模和算法的精华内容，所有基于数据量化认知的模型归纳方式几乎都是这样去做的。说到底，任何数学建模最后落实到算法哲学上，都是想尽办法让模型中的待定因素向着减小误差的方向移动，力求模型的结果和观测到的现象一致。换言之，模型构建的流程也是非常固定的，这是一种被普遍认可的推敲模型的主流且科学的方式。

15.5　本章小结

在本章中，我们讨论了数据建模环节的套路。这个套路是非常固定的，也符合人类认知世界最原本、最朴素的模式，经得起推敲和验证，因此才会成为一种标准的、科学的、系统的建模方式。

在本章的最后，我只说一个问题——关于模型精度的问题。书中提到的模型大都是比较经典的数据挖掘模型，但还有很多前人研究出来的经典模型我们没有展开讨论。以回归模型为例，还有多项式回归、岭回归、COX 回归、泊松回归等。

在很多单位里，对业务数据进行研究的时候也需要建模，这个建模的过程从科学的角度来说也要遵循上面所说的原则。然而，由于各种原因，即便使用了上述方法对数据进行加工处理并拟合出一个模型，也总会出现误差。出现误差时，有的人可能会非常惊慌，认为模型是错误的，并急于从各个可以处理的环节对模型进行优化，试图减小误差。其实我们可以想想看，误差是客观存在的东西，不论测量手段的精度多高，都会或多或少地产生一个与“真实”值有差距的误差值。误差无法消灭，况且在这种惊慌地消灭误差的过程中，也存在发生过拟合的风险。在这种情况下，我们不妨以如下两点作为准则。

第一，在验证集数据上的召回率和准确率不比训练集低。这个说法等同于在验证集数据上的误差不比训练集低。

第二，只要误差值满足业务需要就可以了。过高地追求精确性会提高建模成本，同样有可能得不偿失。

第 16 章　数据可视化与分析

最后一环，是大部分程序员觉得最没有技术含量却又最重要的一环——可视化。数据可视化，说容易也容易，说困难也困难。容易是说，Photoshop、Excel 等工具的存在给广大数据可视化工程师带来了福音，工作量大大减少；困难是说，人类习惯接受的视觉信息一般是三维的，要想用二维的显示介质表现三维的物质，就必须借助类似“旋转”的功能了，如果要表现四维的物质，就必须使用“时间轴”这样的工具了——五维甚至以上该怎么办？数据分析中有几十个维度参与的情况比比皆是。这是第一个难题，除此之外还有，我们一个一个来看。

16.1　看得见，摸得着

在可视化和数据分析这个环节，越来越多的企业注意到，在将报表展现给阅读者的时候，能够进行自由的交互是一件非常重要的事情。所谓“交互”，就一定不是将一张“死”的报表呈现出来，以只读的方式让人被动接受。报表需要承担很多的能动性分析作用，这一点现在很多企业已经掌握得很好了。目前，如果哪个企业的报表不具备足够多的交互功能，那么这个企业很可能没有把信息化工作做到位，更谈不上做好分析工作了——因为工具还不够完善。

传统的报表通常确实只有“报告”的功能，阅读者不能与其互动。不过，现在的报表大都是这种“看得见，摸得着”的报表。我们尝试总结一下报表都有哪些互动功能及其好处。

16.2　颜色很重要

人的眼睛具有识别颜色的特性，不同的颜色对于人的神经刺激作用不同。在报表设计中，颜色也有相应的含义，这与美学研究的范畴相通。颜色可以给人很强的心理暗示，或者表达文字所不能体现的额外的信息。

红色，表示喜庆、危险、热情、暴力等，能够引起人的亢奋或冲动。通常希望引起人们的警觉或者需要人们特别注意的地方会使用红色。例如，红头文件通常都是以红色作为文件的标题颜色，以引起人们的注意（如图 16–1 所示）。再如，在报表中通常用红色来表示告警信息，这同样是为了引起人们的注意（如图 16–2 所示）。

XXXX有限公司文件

×字〔201×〕×号

关于××××的××

图 16–1　红头文件

Displaying 1 to 10 of 10　More...　Search

	Student ID	Last Name	First Name	Overall Score	Overall Label	Attendance	Failures	Incidents
	1	One	Student	55	Moderate Risk	Moderate Risk	Moderate Risk	Moderate Risk
	10	Ten	Student	90	No Risk	Moderate Risk	No Risk	No Risk
	2	Two	Student	70	Minimal Risk	Extreme Risk	No Risk	No Risk
	3	Three	Student	95	No Risk	No Risk	No Risk	Moderate Risk
	4	Four	Student	90	No Risk	No Risk	Moderate Risk	No Risk
	5	Five	Student	100	No Risk	No Risk	No Risk	No Risk
	6	Six	Student	15	Extreme Risk	Extreme Risk	Extreme Risk	Moderate Risk
	7	Seven	Student	75	Minimal Risk	Moderate Risk	No Risk	Moderate Risk
	8	Eight	Student	40	Moderate Risk	Moderate Risk	Moderate Risk	Extreme Risk
	9	Nine	Student	15	Extreme Risk	Extreme Risk	Extreme Risk	Moderate Risk

Show 50　← Previous 1 Next →

Add to Intervention Group

图 16–2　报表中的告警信息

黄色表示温和、警示、友好等。相对于红色来说，黄色或橙色也有一定的警示作用，只是程度要弱很多。报表中通常也有黄色的信息，用于表示程度相对较低的警示含义。

绿色表示安全、生命、和平、自然、稳定等，在报表中常用于描述健康、正常之类的信息。在如图 16–2 所示的报表中，标有“No Risk”（无风险状态）字样的单元格就是用绿色表示的。

蓝色表示沉稳、忠诚、安全、整洁等，在报表中常用来描述相对平和、稳重的信息，不包含过多的感情色彩（如图 16–3 所示）。蓝色属于“百搭”色，而且人阅读久了也不会觉得疲劳或者烦躁。但是，黄色或者红色在阅读时间过长的时候会给人一些不安的感觉。

黑色表示严肃、沉重、酷炫、现代感、科技感等。使用黑色的报表大多数情况下是为了营造一种酷炫或者科技感的意境（如图 16–4 和图 16–5 所示）。而且，黑色系的页面加上高光效果，通常会有硬塑料、玻璃的质感，也容易给人酷炫、高档的感觉。但是，并非每个人都会对黑色产生共鸣，有些人可能非常反感这种颜色，认为它太过晦暗或者压抑，对这种情况我们也需要注意。

白色表示纯洁、高雅等。白色在报表中的使用是最自然的，应该说，白色也是“百搭”色。由于通常使用的纸张就是白色，人们对出现在白色底色上的文字或者图表信息一般不会感到意

外，也大都会接受。所以，如果绞尽脑汁也想不到用什么颜色来做报表的底色，白色是一种非常好的选择。

图 16–3　蓝色报表

图 16–4　黑色报表（1）

图 16–5　黑色报表（2）

16.3　别说布局没有用

16.3.1　由上而下，由简而繁

在数据可视化这个环节，除了颜色以外，布局对数据的展示也非常重要。

在谷歌搜索的结果页面中可以发现，左上角是被点击最多的位置，点击次数向右、向下逐级递减（如图 16–6 所示）。采用这种布局的原因主要有两个方面：一是搜索引擎搜索到的结果，相关性通常从上到下逐级递减；二是人有从上到下、从左到右的阅读习惯。可以说，这种布局完全由是人本身的阅读习惯决定的。所以，制作报表时，通常会将重要的内容放在顶端和左上角的位置，让人一目了然（尤其是报表的首页）。

图 16-6　谷歌搜索页点击热力图

如图 16-7 所示，这种在报表首页顶端直接展现最重要的指标信息的布局，是目前众多以运营人员为对象的报表系统最喜欢使用的结构，整体数字指标一目了然，近期的趋势在下方体现。这种报表可以让阅读者在几秒内把握全局数据，是一种在阅读报表方面时间投入产出比很高的方式。在每个指标上还有“read full report”按钮，这是一种维度下钻的应用。对于感兴趣的数据指标主题，阅读者可以通过这个按钮来了解其详细内容。这种放置方式体现的原则叫作“线索一致”，通常是指将操作或者思路连贯性较强的按钮放在一起。这种方式不仅能降低切换成本，而且不会给阅读者的思维造成“颠簸感”。

图 16-7　报表首页

16.3.2　总-分，分-总，总-分-总

“总-分”、“分-总”、“总-分-总”结构同样是报表中习惯使用的结构。这些方式看起来非常像小学作文课上老师反复强调的写作习惯——没错，还是因为人的阅读和理解习惯而产生的迎合问题。

如图 16–8 所示，总–分结构报表的核心是表格形式的内容，一般是为了展示类似“流水”的信息。因为信息太多，阅读难度比较大，所以通常会添加“Summary”（也就是“概要”或者“小结”）来引导阅读。

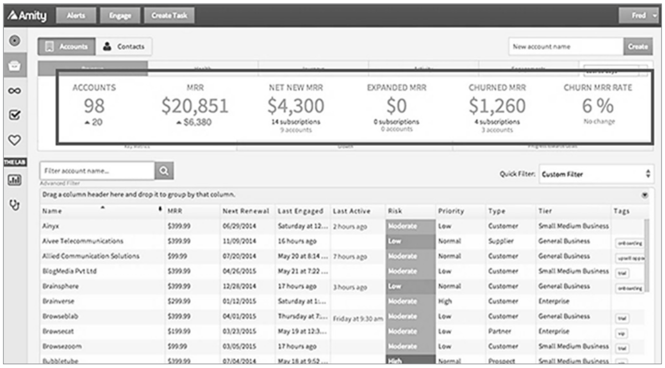


图 16–8 总–分结构

在报表首部加入“概要”信息也会帮助人以最快的方式了解下面内容的大致情况，例如“合计值”、“平均值”、“同比/环比增长值”等都可以作为“概要”的首选信息；而在表格的最后，同样可以放置一行信息作为“小结”，这也是由人的阅读习惯决定的——放在表格最后的汇总信息通常让人印象深刻（如图 16–9 所示）。

	Year								
Month	2007	2008	2009	2010	2011	2012	2013	2014	2015
Jan	6.77	4.24	6.80	5.20	8.60	4.24	6.52	4.77	6.80*
Feb	1.19	2.47	1.53	3.69	2.84	4.50	2.40	3.21*	3.37*
Mar	0.77	3.81	3.79	4.29	4.61	5.66	6.96	8.69*	5.96*
Apr	0.04	1.86	0.09	2.60	5.74	3.72	5.59	3.25	2.01*
May	1.41	2.07	0.04	3.99	2.84	0.74	3.99	4.36	
Jun	2.34	2.55	0.58	2.07	1.01	2.48	0.08	1.13*	
Jul	1.48	0.53	1.13	0.02	2.21	1.61	0.10	1.34*	
Aug	0.55	4.20	2.03	1.13	0.57	0.06	0.98	0.88*	
Sep	1.80	0.81	1.87	5.63	2.32	0.00	5.28	3.43*	
Oct	4.09	2.86	7.07	2.07	3.07	9.22	2.23	7.15*	
Nov	2.92	8.81	8.00	2.27	4.92	7.51	5.30	6.22*	
Dec	7.40	3.71	1.42	4.44	2.80	6.65	2.98	6.58*	
Total (year)	*30.76*	*37.92*	*34.35*	*37.4*	*41.53*	*46.39*	*42.41*	*51.01*	*18.14*
	2007-08	2008-09	2009-10	2010-11	2011-12	2012-13	2013-14	2014-15	2015-16
Total (Oct-Mar)		*24.93*	*27.50*	*29.67*	*24.83*	*25.19*	*39.26*	*27.18*	*36.08*

图 16–9 分–总结构

16.3.3 毗邻吸引

“毗邻吸引”这个词用于形容一种现象——将主题、内容、目的等方面相关性较强的事物放在相近的位置通常会引发更好的共鸣效果和转化率。这种现象在生活中很常见。老百姓有一句俗话叫“扎堆火”，意思就是相同类型的事物放在一起就会显得比较繁荣。我们在生活中会看到很多根据这种“毗邻吸引”原则所构筑的商业体系，例如购物中心、乐器一条街、美食城、家居广场等商业形态。为什么要这样配置呢？这与每个消费者的消费时间均摊成本有关。

以乐器一条街和家居广场为例。购买乐器和家居用品对一个人来说通常是小概率事件——毕

竟购买的不是快速消费品[117]。所以，如果店铺分散，就意味着商家要尽可能提供完整的供货体系才能满足消费者形形色色的需求，而这对商家来说是非常大的负担。如果提供的商品种类单一，通常消费者会因为担心可能买不到需要的东西而打消前往店铺购物的念头。但是，如果把这些商家集中在一起，消费者的心理就会发生变化，因为他们挑选的余地变大了，即使一个店铺无法提供自己需要的商品，还有很多店铺可以选择，买到心仪商品的概率就增加了。从商家的角度来说，同样可以只专注于销售自己了解的货品，而且由于客户流量的增加，会在成交概率不变的情况下，成交量会有所上升。

对于购物中心来说，道理也是一样的，将服装、鞋帽、箱包，以及餐厅、电影院等项目放在一起经营，可以满足消费者一次休闲出行的多种需求。对一个消费者来说，前往购物中心会降低均摊在购物环节及出行环节的时间成本和金钱成本，这种方式显然有明显的成本优势。这就是毗邻吸引原则在商业活动中的体现。

在报表中同样可以使用这种方式来降低阅读者的阅读成本，如图 16-10 所示。

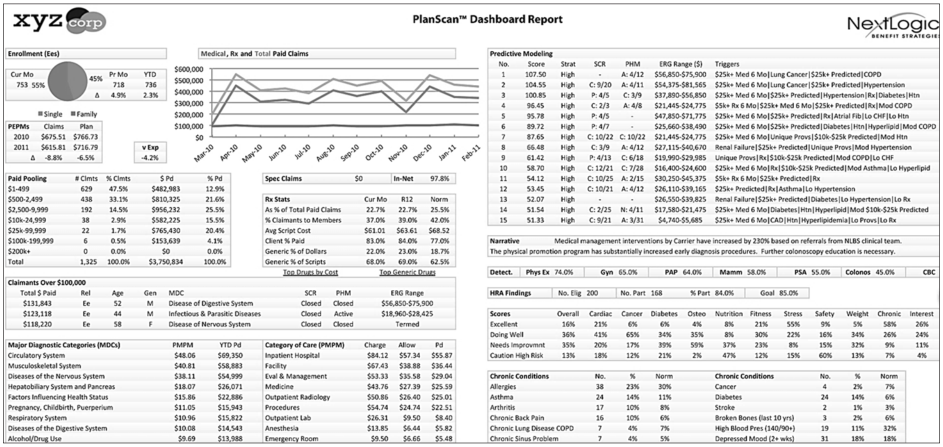


图 16-10　毗邻吸引原则

这张报表中有很多内容。从结构上看，左侧大都是财务相关信息，包括登记人数、支付值、申请金额、各种药品的分类金额等；右侧则是与诊断相关的内容，有预测模型、记述信息、健康风险信息、评分值、慢性病症状等。对于内容较多的报表来说，毗邻吸引原则尤为重要，因为这时需要用若干个板块分别陈述某个方面侧重的信息，再在某个信息分类里展现相关的详细信息，这与写文章时要分章节的思路是一样的。

117　快速消费品（Fast Moving Consumer Goods，FMCG）是指使用寿命较短、消费速度较快的消费品。

16.4 有图就别要表格

美国心理学家乔治·A·米勒[118]在 1956 年发表的论文 *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information* 中，提出了著名的神奇数字“7 ± 2”理论。

这个理论有一个很重要的结论——健康的人记忆广度为 7 个单位。在没有任何提示或者帮助的情况下，人可以轻松记住 7 个杂乱无章的信息单元（例如汉字、字母、数字等）。一些记忆力比较好的人可以记住 9 个，记忆力差一点人的也可以记住 5 个这种杂乱无章的信息单元。这就是我国古代七言绝句和五言绝句比较流行的心理学和数据科学理论解释。甚至像曾国藩[119]这样的治世能臣，在统兵打仗的时候，为了让士兵（尤其是那些文化水平低下的底层湘军士兵）牢记军纪，写下了《水军得胜歌》。

三军听我苦口说，教你水战真秘诀。
第一船上要洁净，全仗神灵保性命。
早晚烧香扫灰尘，敬奉江神与火神。
第二湾船要稀松，时时防火又防风。
打仗也要去得稀，切莫拥挤吃大亏。
第三军器要整齐，船板莫沾半点泥。
牛皮圈子挂桨柱，打湿水絮封药箱。
第四军中要肃静，大喊大叫须严禁。
半夜惊营莫急躁，探听贼情莫乱报。
第五打仗不要慌，老手心中有主张。
若是好汉打得进，越近贼船越有劲。
第六水师要演操，兼习长矛并短刀。
荡桨要快舵要稳，不慌不忙听号令。
第七不可抢贼赃，怕他来杀回马枪。
又怕暗中藏火药，未曾得财先受伤。
第八水师莫上岸，止许一人当买办。
其余个个要守船，不可半步走河沿。
八条句句值千金，你们牢牢记在心。

118 乔治·A·米勒（George Armitage Miller，1920 年 2 月 ~ 2012 年 7 月），美国普林斯顿大学心理学教授，著名心理学家。

119 曾国藩（1811 年 11 月 ~ 1872 年 3 月），晚清重臣，近代政治家、战略家、理学家、文学家。

不要笑，这种朗朗上口的方式是最便于普及和记忆的，这是一种韵律性的信息“可视化”。

人自身有一种非常致命的、难以克服的缺陷——同时容纳的信息量太小。例如，我们在同时和两个人对话的时候，只能让他们一个一个地说，如果这两个人一起开口，我们就很难分辨他们各自说的是什么。这种现象同样存在于视觉、嗅觉等感官领域。所以，在对数据进行分析之后，得到的可视化内容一定要简洁明了，这一点非常重要。

假如一个完整展现在人眼前的画面，其内容全部都是文字，如果想从中获取信息，就需要把这些文字从头到尾读一遍，然后归纳总结出一个主题思想。所以，用长文字进行描述的方式看上去内容很丰富，但是提取信息的效率却很低。

可视化的价值就在于帮助阅读者抓住重点信息。在“抓住重点信息”这个任务中，可视化工程师要和阅读者一起完成工作。可视化工程师下的功夫越多，信息提炼越到位，阅读者的阅读效率就越高，可视化工作的意义就越重大。

向上一层，就是通过大标题和小标题来归纳，并附带相关详细信息。这种方式在新闻类信息可视化工作里用得最多，效果还是比较好的。

再向上一层，就是纯粹用条目性解析的形式来表达。这种方式是大标题和小标题的变种，略去了附带的详细信息，优点是在一个画面中对核心内容的聚焦效果更好。

再向上一层，就是利用表格对信息内容进行归纳和对比。表格这种在二维空间中天然具有维度划分功能的形式非常适合用来进行对比，对内容的罗列和陈述也很有帮助。

再向上一层，就是图表。图表表现的内容通常最少，特点也最明显——维度少，数值化，更容易让人对对象概念聚焦。说这种形式好，还有一个根本原因，那就是人类获取信息的特点。人类的双眼对图形和图像的敏感程度比较高，对这些信息的获取比较直接。而在面对文字这类符号信息的时候，人类先要将它们读取，传送至大脑，经过理解映射成具体的事物及量化关系，这个过程消耗的时间比直接读取图表信息的反射时间要长，具象性弱了很多，视觉冲击也不够强。

所以，对于一段需要陈述的内容，表现形式的优先选择顺序通常是：图表、表格、条目、标题加详情文字、文字段落。

16.5　分析的内涵

数据分析工作在不同的单位是由不同岗位、不同职种的人员完成的，有的分析工作是在做统计的时候就已经嵌入算法当中了，而有的分析是在一般性的指标统计呈现在报表或者文件中以后，通过其他辅助分析软件和人的阅读判断来进行。这两类分析工作都是分析，但分析应该放在哪个环节更为妥当，分析工作应该由什么样的人来完成更为合适呢？是由偏重业务（具体场景）的人来做，还是由数学超牛的“大拿”们来做呢？我想，如果我们能够理解分析的内涵和实质，这个问题不难解答。

说到分析，究其本质，就是通过对事物的感知及对比、类比、推测、反证等各种研究方法得到事物规律的过程。既然如此，凡是帮助人们逼近事物规律的认知过程都应该算作分析，或者说，凡是有助于消除不确定性的与数据互动的行为都应该算作分析——我认为这么说绝对不夸张。所以从这个角度来说，程序员在数据库里进行的各种加和值、平均值、方差、数量的计算工作，都应该算是整个分析工作的一部分。不过，通常没有人把这些已经得到广泛应用的认知方式算作分析，因为它太“直接”，没有通过复杂辩证去伪存真的过程——含金量太低。

按照这种“含金量”的说法，目前获得普遍认可的有意义的分析工作是什么呢？从目的出发，大致可以分为以下几种。

16.5.1　相关性分析

相关性分析是所有分析里最常见的一种，也是在日常生产中使用最普遍的一种。这种分析的目的就是通过事件或者数据发生的彼此连带程度来判断其相关程度。如果两个事件大概率前后发生或一起发生，就认为二者有关联性，或者说关联性较高；如果两个事件中一个发生的情况下会大概率抑制另一个的发生，则认为二者是负关联。

在这里我一直用的是“相关”、“关联”这样的词，而没有用“因果”。这也是不少刚入行的分析师容易在逻辑上犯的一个小错误——“相关”是否等于“因果”？答案很明确——不等于。

有这样一个案例（这个案例也出现在《爱上统计学》这本书中）：“在美国中西部的一个小镇，一名地方警察局局长发现了如下神奇的规律，也即冰淇淋销量越高，犯罪率就越高；反之，冰淇淋销量下降后，犯罪率也跟着下降了。城市议员 Joe Bob 得知上述规律后，便兴奋地提出了他的政见，也即‘限制冰淇淋消费以便降低犯罪率’。朋友们应该能察觉，这想法甚是荒谬，可问题出在何处？原来，当气温逐步变暖后，会呈现白昼更长、门窗常开、衣着较少等现象，进而引发犯罪率的提升。不巧的是，冰淇淋销量也往往随气温同步变化。于是，两个本无因果关系的事件，却因数据上的极大相关性而凑在一块儿，被生硬解读后，冰淇淋也终于‘中枪’倒地。”

这类错把“相关”当成“因果”的事情其实用归谬的方法也能很快得到验证。在实践中，简单地说，就是把被认为是因果关系的两件事单独罗列出来，然后靠增大（提高）或降低（减小）“因”事件的程度来观察“果”事件对应的程度。当然，最好能够用量化的方式去观察。

在这种目的下，常用的分析方法很多，例如朴素贝叶斯算法、贝叶斯信念网络、Apriori 算法等。这类基于统计的概率模型通常能够比较好地呈现关联关系，为我们的分析起到了很好的过滤作用。

朴素贝叶斯算法是一种典型的使用量化概率的方式计算事件之间关联程度的算法。在 11.8.2 节我们已经对朴素贝叶斯算法进行了比较详细的介绍，这里只说说对这个模型的理解。

例如，在全局范围内 $P(A)$ 是一个值，但是 $P(A|B)$ 比 $P(A)$ 大，这说明在发生事件 B 的情况

下，事件 A 更容易发生——至少观测到的结果是这样。还要说明一下：即使在这种情况下从数据层面去理解，同样不能下结论说事件 A 和事件 B 有因果关系，而只能说事件 B 与事件 A 正相关（如图 16–11 所示）。相反，如果 $P(A|B)$ 比 $P(A)$ 小，就说明在事件 B 发生的情况下会抑制事件 A 的发生，事件 B 与事件 A 负相关（如图 16–12 所示）。这里说的事件 A 或事件 B 都属于复合事件，例如事件 A 可以定义为多个事件的与或非关系，事件 B 也可以。

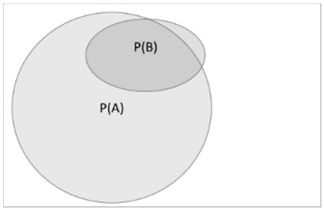


图 16–11　正相关

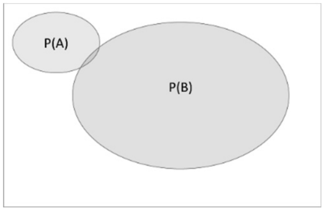


图 16–12　负相关

在任何一个 BI 或者大数据系统里都可以进行“无脑”式的相关性分析工作，并在正负相关性的基础上进行过滤，找到有趣的模式。这种试探性的认知方式使用极为普遍，而且建模容易，数据层面的证据充分，理解难度低，是大数据应用中一种很有效的分析方法。

16.5.2　预测分析

预测分析也是在分析实践中使用比较多的。预测分析通常是对历史性的多维度数据进行建模和拟合，然后通过拟合出来的模型对未来或过去但未知的数据进行推测。

预测分析从目的上来说带有比较明确的量化目标。从数据科学的角度看，预测分析的准确性本身同样是无法自验证的。所以，在使用数据进行验证的过程中，只能选择其中的一部分作为训练数据供拟合使用，剩下的部分则供验证使用。在实践中，训练数据与验证数据通常为 4∶1 或 3∶1（抑或其他比例）。这种验证集存在的目的主要是避免在预测分析过程中产生过拟合现象。这种同样通过量化数据的归纳得到的预测模型，是用验证集数据的召回率和准确率来做评价的，只要召回率和准确率满足场景的需求，就是一个科学的模型，可以用来做预测分析。

在预测分析中，如果预测的目标是连续的数值，通常期望被表达成一个解析解与误差的形式；而如果期望被表达成一个离散的枚举类型，通常期望被表达成一个概率形式。基于这种目的，通常会应用线性回归、逻辑回归、支持向量机、决策树、随机森林、隐马尔可夫模型等算法或者深度学习的相关算法。

这些方法无一例外，都是用“已知”的训练集数据来做训练，把模型训练好之后，再用验证集验算的方法来确定模型的可靠性（主要是验证数据召回率和准确率是否达到业务需求）。召回率和准确率是衡量模型准确性时最关键的指标，也是用数据为模型的可靠性提供证据的唯一标准。不管训练模型的人是什么专业毕业、有什么工作经验、采用什么理论依据，最后只看这

两个指标。进行模型改进的时候，也是看这两个指标是否都有明显提高。

当模型的可靠性满足业务需求以后，就要让其进行预测分析了。这个预测分析就是将一个待预测标签的向量放入模型，使其生成一个标签的过程，通常是对一个未知的未来状态的预计值。

16.5.3 其他分析

在日常生产中，相关性分析和预测分析作为两种主要的分析目的，涵盖了大多数分析场景。这些分析的结果通常可以直接用来指导生产，相对而言，指导的目的比较清晰，要么解释或找到隐藏的因果关系及关联关系，要么通过模型来进行一定的预判和猜测。

除此以外的分析目的则相对是泛泛而谈，没有想要找到或解决的具体问题和确切的认知目标。这种情况下会使用一些无监督学习或者半监督学习的机器学习模型进行处理，例如聚类、半监督回归等。这类学习通常以试探为主，在分析前对分析的结果不具有明确的倾向性和期望。

如果你觉得这种说法不是很明确，那么我们来看一些统计应用中比较有趣的案例。

16.6 有趣的统计应用

在和很多分析人员攀谈的过程中，我听到过一些抱怨：“这些数字完全是随机的，太糟糕了。”“随机的东西没什么价值。”不过，随机真的不好吗？

人们对客观事物认知的不断深入是一个熵减的过程。人们厌恶信息熵，总觉得它意味着杂乱、未知，是千百年来都要消灭的认知“大敌”。降低认知中的熵也是数据科学的使命中最为重要的组成部分。熵和随机不分家。在相同条件下重复进行试验，每次的结果未必相同——这种困扰对于认知能力有限的人类来说永远存在。然而，基于“尺有所短，寸有所长”的辩证逻辑观点，“随机没有价值”这样的观点是值得怀疑的。著名统计学家 R.C.Rao 的 *Statistics and Truth* 一书中有一些案例就很好地利用了随机这种特性的优点。

16.6.1 不规则图形的面积

平面中有一个不规则图形（如图 16–13 所示），我们能求出它的面积吗？乍一看用初等数学中的割补法恐怕难有进展，所以不妨把它放到计算机中，用统计学中的概率来尝试计算。

随机产生一个平均分布的序列 x，取值范围为 1 到 n，n 为整个矩形的宽度像素数。随机产生一个平均分布的序列 y，取值范围为 1 到 m，m 为整个矩形的高度像素数。在大量产生向量 $a_i = (x, y)$ 坐标值的过程中，由于 x 和 y 是平均分布的，所以落入图形中的概率值与整个矩形的概率值之比就是其面积。在 i 无限趋向 ∞ 的过程中，这个比值也在无限接近理想的面积 S。

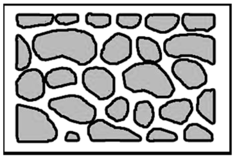


图 16–13 不规则图形

如果能够获得这些图形边线的方程当然很好。如果不能，也可以用颜色差异来判断生成的 a_i 向量落入不规则图形与未落入不规则图形的区别。这就是一个巧妙利用随机求不规则图形的数值解的方法，而且可以推广到无限高维。

16.6.2　套出你的实话

在以访谈的形式获取调查数据的过程中，可能会遇到一些尴尬，例如问题可能触及受访者的隐私。即使调查者信誓旦旦地对受访者保证绝不会外泄信息，受访者还是会心存芥蒂，答题的时候很可能会给出一个“不正确”的答案。

如果在问卷中明确写着“您吸食大麻吗”，就算是不记名问卷，也会有相当比例的受访者因为畏惧交卷时被他人看到自己的答案流露出不屑或鄙夷而填写“否”。怎么办？这时可以和受访者做一个“小游戏”。凡是涉及这种情况时，都可以使用一个“无害”的问题作为配伍。

一个问题被设计成了两个问句，一个是 S（您吸食大麻吗？），另一个是 T（您的手机号码尾数是偶数吗？）。受访者被要求掷一次硬币，如果正面朝上，就回答 S，如果反面朝上，就回答 T，而且不用标注究竟掷出来的是哪一种情况。这下气氛就轻松多了。在实验中，大部分受访者都会老老实实掷一次硬币，然后根据硬币的正反面来回答对应的问题。

假设我们要求的吸食大麻的人数比例为 π，手机尾号为偶数的人数比例为 λ，该题目回答“是”的人数比例为 p。根据随机的原理，回答 S 题目的人数比例约为 $\frac{1}{2}$，回答 T 题目的人数比例约为 $\frac{1}{2}$。在选择 S 题目的人中，回答“是”的人数比例为 $\frac{\pi}{2}$，回答“否”的人数比例为 $\frac{1}{2}-\frac{\pi}{2}$；在选择 T 题目的人中，回答“偶数”的人数比例为 $\frac{\lambda}{2}$，回答“不为偶数”的人数比例为 $\frac{1}{2}-\frac{\lambda}{2}$。所以，$\frac{\lambda}{2}+\frac{\pi}{2}=p$，用 $\pi=2p-\lambda$ 来估算吸食大麻的人数比例。其中，p 是统计结果中的已知值，λ 则可以要求受访者留下手机号码或者干脆在大样本中直接取 $\lambda=\frac{1}{2}$。

这里使用的同样是随机现象所产生的正面影响。是不是很有趣呢？

16.6.3　巧测圆周率

图 16–14　蒲丰

“蒲丰投针实验”是由法国数学家蒲丰[120]（如图 16–14 所示）完成的一个有趣的数学实验。

取一张白纸，在上面画许多条间距为 d 的平行线。取一根针，长度为

120　蒲丰（George–Louis Leclerc de Buffon，1707 年 9 月 ~ 1788 年 4 月），法国数学家、自然科学家。

$l(l \le d)$。将这根针在这张纸的上方向纸上随机投掷 n 次，记录每次针与直线相交的次数 m，可以得到一个很神奇的结果：

$$\lim_{n\to\infty}\frac{2l}{d}\cdot\frac{n}{m}=\pi$$

1901 年，一个名叫拉泽里尼的人在投掷 3408 次后得到了 $\pi\approx 3.1415929$ 的小数点后 6 位的精确数字。在这个试验中没有圆形出现，而且投掷的动作是随机的，可是 π 为什么能够通过这种方式测量出来呢？这个看似简单的现象背后其实也是一系列具体、扎实的数学逻辑证明。

如图 16–15 所示，设针与距离其最近的直线的夹角为 θ。取针的中点，并以该中点向距离其最近的这条直线做垂线段，长度为 x，则有

$$x=\frac{l}{2}\cdot\sin\theta$$

当 $x\le\frac{l}{2}\cdot\sin\theta$ 时就会相交。由于针的位置是随机的，所以 x 在 $(0,\frac{d}{2})$ 上服从均匀分布，θ 在 $(0,\frac{\pi}{2})$ 上也服从均匀分布。这个时候的 π 是未知数，是根据弧度的定义写出来的。这里我们又一次看到了求不规则图形面积的思路。如果想求曲线

$$x=\frac{l}{2}\cdot\sin\theta$$

在平面直角坐标系 (θ,x) 上的面积，那么仍然是一个产生随机点求个数比的问题。

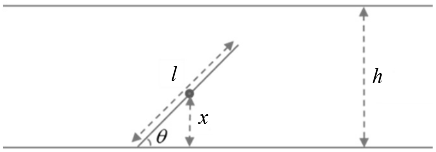


图 16–15　相交的情况

如图 16–16 所示，阴影部分与整个面积的比例 p 可以写作

$$p=\frac{\frac{1}{2}\int_0^{\pi}\sin\theta d\theta}{\frac{d}{2}\pi}=\frac{2l}{\pi d}\approx\frac{m}{n}$$

所以得到

$$\pi\approx\frac{2l}{d}\cdot\frac{n}{m}$$

真是精巧绝伦。

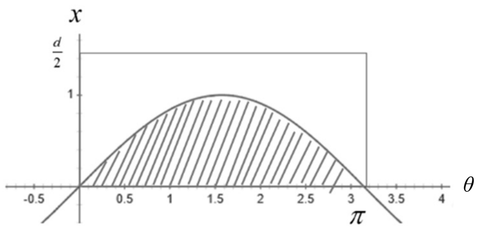


图 16–16　求解面积问题

在本节的案例中，一些利用统计原理来做的工作看似简单，却蕴含着深刻的数学原理。这些既利用随机原理又能得到想定结果的方式，非常值得我们关注。在拥有足够的基础理论知识后，这些技巧使用起来将会如鱼得水。数据分析是一个厚积薄发的工作，关键在于不断积累和总结。

16.7　仁者见仁，智者见智

算法是死的，但人是活的。在使用算法进行计算的时候，人们可以进行参数的设定。计算出来的数据结果，也要靠具备不同知识背景、业务水平和分析能力的分析师来解读。所以，作为一名优秀的分析师，会写算法、了解算法的运算过程仅仅是入门，真正困难的是如何用辩证的观点去看待这些计算出来的数据并进行合理的阐释。

在数据分析环节，我们又一次见到了由人的主观看法大量介入的不可避免的事实。只要是有人参与的环节，都难免会产生“条件相同，但结果不同”的情况。人作为一种极为复杂的“分类器”，其微观行为都是难以捉摸的。所以，即使是同样的数据，不同的人也会分析出不同的结果，进而导向不同的决策——这也不是什么奇怪的事情。我们都听说过，很多大公司，甚至是有上百年历史、上千亿资产的大公司，由于各种原因产生决策失误，最后走向倒闭、被并购等令人叹惋的结局。不要以为这些大公司的数据分析人员和决策人员都是无能的白痴或者骗子，仅仅是因为他们的错误才导致公司走向衰亡。我们必须清楚一点：人类对未来一直都是充满敬畏的，其中重要的原因是未来对人类而言都是不可知的，没人能知道下一刻会发生什么，就更别说 5 年甚至 10 年以后会发生什么了。就好像我们开车去一个遥远而陌生的地方，即使打开导航和 GPS 定位并采用躲避拥堵模式，就肯定能躲开所有的拥堵路段吗？你能确定在途经的某个只能通过一辆车的路口没有突发的事故吗？这个比喻要表达的意思已经非常明确了。

对于数据分析，也只能直面这个悲观的事实：人的认知能力是有限的，对于越远期、越微观的事物，人类的把握能力越弱。从数据分析师的职业道德角度来说，恪守以数据为本的原则，认真处理数据，基于自己最大的认知能力解读，说实话，认真听取其他拥有不同知识背景的分析师的观点——就够了。这是对自己工作的尊重。不必太在意分析是不是面面俱到、会不会出

错——没有人能保证自己的分析与客观世界一致。数据科学只涉及认知边界内的量化知识，如果想尽可能做到一致，那就努力开拓自己的认知边界吧。

16.8 永恒的困惑

做过分析工作的人应该会和我有同感：在做数据解读和分析的时候，有一个永远解不开的困惑，那就是所有的预测和分析都要经过实践的验证才能够被证明或证伪。这就意味着，无论一个人有多么高的算法编写能力或者数学分析能力，在分析一个具体问题的时候，一定要等到事实的发生与其判断的一致，才能认定其正确性。在被认定之前，无论有多少证据或者苗头使判断看上去依据充分，也只能到事后才能认定。

这个结论是非常无奈的。

很多做数据画像的公司或者团队会对其服务的对象信誓旦旦地承诺，说他们的分析是多么精确，画像是如何精准，能够帮助客户大大提高广告投放的准确率。但是从数据科学的角度来说，这一切承诺都是苍白无力的。听者如果是具有数据科学素养的人，也会问同样的问题——这些画像帮助哪些公司提高了转化率？提高的比例是多少？分析拟合的召回率和准确率是多少？ROI[121] 是多少？如果不能给出令人信服的证据，再多的鼓吹都是空中楼阁。

这就好比一位画师给甲画像，然后把画像给乙、丙、丁等人观看，并宣称这幅画像惟妙惟肖、细节逼真。如果乙、丙、丁要判断画师的话是不是正确，评价画师的画功是不是如其所说，要么拿一张甲的照片来对比，要么直接把甲叫到面前来观察，只有这样才能真正用应验性的方法来对画师作出评价。次之的选择就是看这位画师过去画过多少肖像画，人们对其画功的赞誉有多少，将这些内容作为评价先验概率和似然度的条件，通过这种方式来判断“画像惟妙惟肖”是否有很高的置信度。

在数据分析场景中，同样有这样的问题。即使有同样的数据，同样的知识背景，不同的分析师也有可能得出不同的分析结果。哪种结果和事实更接近？没有人能够在最终的事实出现之前下结论。然而，在这种情况下，有着丰富经验和极好“历史战绩”的分析师给出的结论通常更有置信度。但我要强调的是，即使如此，也不能在分析结论被验证之前就说某位分析师的结论一定是对的，这才是困惑所在，也会直接或间接影响决策的实施。下面我们就会讨论决策的实施问题。此外，分析的边界是什么，分析到什么程度才算可以？对于这个问题，不同的流派也持有不同的观点。

我的观点是这样的。在以因果关系为导向的分析场景中，我们所做的事情实际上是在解开一个个“黑盒子”。我们把一个事件 A 当成“因”，把另一个事件 B 当成“果”，那么事件 A 与

121 投资回报率。

事件 B 的关系本身就是一个“黑盒子”。当挖掘得到一定成果，发现了事件 A 和事件 B 的关系后，这个“黑盒子”就被打开，变成了两个更“小”的“黑盒子”，即事件 A' 和事件 B'。然而，一般作为“因”的事件 A' 会更吸引人们的注意。我们会再打开 A' 的“黑盒子”，看看是什么导致了事件 A'——解开后可能是两个更“小”的“黑盒子”。这种逻辑层面的深入研究是无法说清回溯尽头的。通常将分析边界划定在人们目前能够清晰感知的地方就够了，不必探求过深。

举个例子，玻璃棒和丝绸摩擦后，二者都会对轻小物体有吸引作用。通过研究发现：这种摩擦可以使丝绸带负电荷，使玻璃棒带正电荷；带有相同电荷的物体互相排斥，带有不同电荷的物体互相吸引。这种现象就是静电感应现象。带有正电荷的物体甲在与不带电的物体乙靠近时，甲所带的正电荷会排斥乙所带的正电荷，使其向远离甲的方向移动，同时吸引乙所带的负电荷向靠近甲的方向移动，从而形成吸引现象。到这里，第一个“黑盒子”已经解开了。至于为什么电荷同性相斥、异性相吸，则是下一层“黑盒子”。如果还是在经典物理学的范围内，就大可不必往下研究了，或者等到理论基础足够从下一级的层面来解释时再进行研究。也有一些问题确实神奇而精妙，总会让我们感觉到有一个人类在认知上不容易迈过去边界。在这种情况下，也可以不迈过边界——只要当前的理论体系确实足够完善，能够解决眼下的问题就好。

16.9　本章小结

数据可视化与分析是整个大数据落地过程中较为靠后的环节，通常伴随着很多的配色技巧、展示技巧，以及分析人员自身的行业背景和从业知识。应该说，在整个决策制定之前，所有数据层面的产出都应该在这个环节进行充分的呈现。

数据可视化同样不是一个孤立的过程，其中就包括数据与人交互并帮助人认知数据的过程。作为分析的实施者，数据分析人员应该有丰富的行业背景知识，并且要了解数学、地理、物理、心理学等相关学科的知识。分析过程中引入的维度越多，分析的维度就丰富，分析出来的结论通常也就有着越好的视角和说服力。

说到底，数据可视化主要是为了迎合人类自身的阅读习惯，以及人类对颜色、条目的敏感性这些因素而进行的技巧性调整。

第 17 章　数据决策

决策是所有组织及任何一个阶层的领导者都会做的事情，而且几乎天天在做。决策不限于在特定范围或者全体范围内获得重要决议，每个人时时刻刻都在做着决策。

决策是指决定的策略或办法。《韩非子・孤愤》第十一中谈到：“智者决策於愚人，贤士程行於不肖，则贤智之士羞而人主之论悖矣。”意思是说：“智者的计谋由愚蠢的人来评判，贤者的品德由不贤的人来衡量，那么品德好、有才智的人就会感到耻辱，而君主的论断也必然荒谬了。”这里的“决策”显然是指对策略或主张的评判。在人们做出各类行为的时候，实际上也在不断对这个行为的得失进行评价，然后决定是否实行。注意，这个过程不见得是开会，很可能是一瞬间在一个人的脑海中做出了评判，然后实施。

17.1　决策就是“拍脑袋”

决策随处可见。拿着外卖菜单决定午餐的菜色，在大雨中举着伞小心翼翼地走过坑洼不平的道路，在网上看到一件漂亮的衣服然后决定购买，都是在做决策——通过判断方案的好坏来决定是否要实施。这个过程的本质是人类基于自身“价值观”做出的“趋利避害”的行为。这种“价值观”是人类自身的一种对于取舍的判断模型，“利”与“害”是人类自身拥有的对事物感触的结果，通俗地说就是“愉悦”和“痛楚”。

每一个神智正常的人都会具有类似却又各不相同的对“利”与“利益”的理解——有的重视金钱，有的重视名誉，有的重视交际，有的重视短期效益，有的重视长期效益，即使是吃一顿美餐、看一部有趣的小说或电影这种纯粹在情感上获得愉悦的效果，也是“利”最朴素的表现。

与此相反，以上这些事情的对立面就都是“害”或“痛楚”。金钱损失、名誉损失、朋友绝交、看了令人作呕的画面或者被针芒扎到都是“害”。在这些事情中，凡是已经几乎形成条件反射或非条件反射（例如脊柱反射）的，就无须通过一定的对比、分析、推理、判断来做决定。虽然本质上都是“趋利避害”，但这些事情通常不属于我们刚刚所说的“决策”。而根据“利”与“害”的直接程度和大小，有时可以在瞬间做出决策，有时则需要相对复杂的决策过程。对简单的、可以瞬间做出的决策，“拍脑袋”就够了，因为由其结果造成的“利”与“害”都微乎其微。

然而，最为复杂，也是本章重点描述的，是那些影响相对较大、较深远的，或者对利益有重大影响的决策。这些决策通常值得多花一些时间和金钱成本来做出。这种决策有方法论可循吗？也是有的。

17.2　哪里有物质，哪里就有数据

17.2.1　目的的统一

前面我们已经对决策评价进行了一些讨论，其中一个鲜明而重要的观点就是：决策的目的一定要统一或一致，否则决策会是一个混乱、漫长且令人充满疑惑的过程。换言之，决策的问题如果能最终转换成在多个方案之间比较多个没有二义性的数字大小的问题，那就非常完美了。这里有一个概念叫作“评价函数”。

所谓“评价函数”，就是类似于

$$value = f(\vec{x}) = f(x_1, x_2, ..., x_n)$$

形式的函数。*value* 就是方案最终的评价或者价值，通常是一个正实数，这样便于参与者讨论及理解。$x_1, x_2, \cdots, x_n$ 的内容是这个评价体系需要考虑的所有因素的量化值。$f(\vec{x})$ 是这些被考虑因素的计算逻辑，内容可以很复杂，甚至在有些比较复杂的场景中，仅 $f(\vec{x})$ 就能写满几页纸。

这样，一旦最后所有人对 $value = f(\vec{x}) = f(x_1, x_2, ..., x_n)$ 的解释没有异议，对其结果就应该不会有异议。当然，$\vec{x}$ 中 $x_1, x_2, \cdots, x_n$ 的来源应该有足够满足当前决策需要的可信性。

在一次决策的过程中，一定有人对决策过程或者决策结果不满意，所以决策过程中的不满意程度往往会成为一个非常简单易用的判断标准。这种决策方式我们都见过并亲身经历过。例如，举手表决，少数服从多数。这种简单的一人一票制度是一种典型的以“决策过程中的不满意比例（人数）”最低为决策标准的方式。看，这种我们习以为常的决策方式也有理论依据。

中国有句俗话，“宁跟明白人吵架，不跟糊涂人讲话”。这句话的隐藏含义就是：和所谓的“明白人”吵架是一件能够得到结果的事情。这里说的“明白人”是指那些利益观点、是非观点和自己一致的人，吵架的原因也是视角不同或者考虑不周全，通过争论可以对事情进行改善或弥补。“不跟糊涂人讲话”说的则是那些目标观点不明确或者与自己的目标体系相差太远、无法调和的情况。在有的场合我们会发现争论双方都觉得对方糊涂、不可理喻，原因就是双方目标体系相差太大，无法就一个评价目的进行讨论，这种讨论也很难有结果。

在董事会这种用股票赋予不同量的决策权重的场景中，情况略有不同。每个人不再被完全平等地赋予同样的对满意或不满的表达量，而是按照其持有的股票数量分配决策权重。为什么这样做呢？原因也非常简单：股票代表利益，每一股所代表的利益是均等的，持股越多，能够获得的利益就越多。董事会的决策通常是在整个企业范围内起作用的，影响较为深远，涉及的

利益也要覆盖每一位股东。这种以所持股票数量来表达观点的方式是一种加权方式，其好处是能够在利益主权至上的前提下，在全局范围内保证“利益主张”最大化。

17.2.2 数据胜于雄辩

现在，绝大多数企业，尤其是中小型企业，在做决策的时候往往是几个主要负责人坐在一起“拍脑袋”，能够依靠数据进行决策的凤毛麟角。为什么呢？因为数据决策的成本目前仍然比较高，实施的难度也比较大。

这是一个客观现象。在整个“实践篇”，我们用大量篇幅描述了完整的数据生命周期，从数据的生成和收集，一直到数据的可视化和分析，整个过程非专业人员很难较好地完成。对很多中小型企业来说，自身的生存状况还不允许其每位领导者都采用这样“完美”的方式去完成决策——成本是硬伤。

不过，我还是强烈建议，即使是几个主要负责人在一起“拍脑袋”，如果想让方案更有说服力，仍然要有数据或数字作为观点的支撑。用数字说话有如下好处。

第一，直观。数字是不需要人们再去想象的东西，该是多少，就是多少。如果有图表作为辅助那就更好了，其好处在前面我们不止一次讲过——加强视觉冲击，从而加深印象。

第二，清晰、准确。数字的表达具有准确性，其程度描述是非常清晰的。如果不使用数字，在讨论问题的时候应该如何阐述结果？“很多”、“不太多”、“差不多挺多”、“非常多”、“特别多”——这几个词表述的内容到底哪个最多？具体有多少？基于这种表述的内容显得苍白无力，讨论也无法继续下去。

第三，改进方便。改进方便有两方面的含义。一方面，经过一段时间的决策实施后可以观察相关数字的变化，例如一些关键指标值。如果改观程度较大或者符合预期，就说明决策有效；反之，就要重新审视其中的问题。另一方面，这些数字可以进行组合与叠加，形成丰富多样的 $value = f(\vec{x})$，这为评价方式的改进提供了绝好的数据基础。

在数据决策中很多朋友会有困惑，其中之一就是：我的数据不太准确，决策会不会出问题？我的观点是：数据准确比不准确强，不准确比没有强，只要不是完全没有依据或者故意误导决策即可。而且，不同的统计来源或统计规则所产生的数据，即使有同样的标题，也不要轻易拿来做对比。这种“刻舟求剑”式的错误不应该出现在现代企业中智者的大脑里。

17.3 这是风险博弈

说到底，决策永远都是风险博弈的问题。所谓风险博弈，用最简单的视角去看，只有两个因素——“得”与“失”。

这里说的“得”与“失”，可以和前面所说的“利”与“害”分别做同义解释。决策无非就

是权衡这些“得”与“失”、“利”与“害”的关系。从前面讨论过的哲学观点去忖度，很快就可以看到“利”与“害”之间的转化是多么方便。

17.3.1 性价比优先

在投资领域有一个专业的缩写名词“ROI”，即“投资回报率”（Return on Investment）。ROI以一个时间周期为单位，用投资回报的资本数量除以投资金额。这种方式很容易被人理解。例如，我们平时在银行或者理财机构看到的诸如“某基金产品，投资回报率为年化 7.5%”，意思就是：如果投资 100 万元，那么一年后将拿到本金 100 万元再加上利息 7.5 万元。有了这种方式，在做投资决策的时候，人们都会比较不同金融产品的 ROI，哪个高就投资哪个。不过有些投资机构很不厚道，在宣传中隐去很多细节信息和复杂的兑现条件，相信不少人都吃过亏。

ROI 这种方式不仅在投资领域有应用，在其他企业的决策中也常常会用到。例如，广告商会向广告主许诺 ROI 的提升指标。在投放某种广告的情况下，花费为 I，收益为 R，ROI 的计算方法是：

$$ROI = \frac{R}{I} \cdot 100\%$$

其他企业同样有这样的账可以算。例如，举办一次企业宣讲活动，花费经费为 I，期望的 R 是在此后一周内销量提升的部分，也就是活动举办后一周的销量总额与活动举办前一周的销量总额的环比差值，同样可以使用这种方式进行计算。这种账很好算，而且可以应用于所有可以评价出花费 I 和收益 R 的场景。

相信细心的读者朋友可能会发现，这里的分母是 I。那么，如果 I 特别小，岂不是收益巨大？甚至，如果 I 的值是 0，那么 ROI 是不是就是理想中的无穷大了？现实并不是这样啊。

没错，刚刚这些例子中，我们说的花费 I 都是割裂的。一个公司或一个团体的成本计算不会采用这样的方式。如果要计算，I 中要包括平摊成本及时间成本。

为了把这个模型叙述清楚，我们要严谨地对其进行改造：

$$ROI = \frac{R}{I \cdot T} \cdot 100\%$$

R 是由本次决策带来的直接收益，要注意用类似刚才的环比差值的形式减去这次计算中的“背景噪声”。

I 是本次决策的投资额。这个投资额是指广义的投资额，包括人员成本、房租水电费用、设备折旧、公关费用、物品采买费用等。注意，在比较不同决策的 ROI 时，包含的因素要相同。

T 是指事件作用的时间范围，这个时间范围在实践中可能不太容易把握。如果是做季度预算或者年度预算，T 是一定的，直接取 90 天或者一个自然年（财政年也可以）的数据即可。如果是针对某一次活动，例如刚才说的一次宣讲活动或者一次广告投放活动，一般来说要借助行

业经验来判断长尾通常会持续多久——只要做过一两次，很快就会有一定的判断能力，所以完全没有必要担心。这里的 R 和 I 对应的就是 T 涉及的收益和花费。

当这些值确定后，就可以计算 ROI，并比较不同方案、不同周期 ROI 的大小了，一般来说是越大越好。

17.3.2　小迭代至上

在对 ROI 的评价中有一个重要的原则就是“小迭代至上”，也就是说，通常 T 不要取太大。如果 T 太大，就意味着这个决策影响的周期很长，而在这段时间内会发生很多未知的事情，这些事情或直接或间接地影响着 R 和 I 的预估准确性。而且，即使严格地控制 I 的大小，T 也会一刻不停地向前流动，哪怕是在开会时间发生的成本也要计入 I。

在大型企业的决策中，由于 I 里面包含的决策成本过高（通常由高层管理人员参加，中低层管理人员进行相应的数据准备与配合），因此决策本身消耗的人员成本都不能忽略，而是要包含在 I 中。这种情况下，如果 T 取值太小，就会让 I 的比例提高。所以，为了提高决策的 ROI，在公司层面通常都是季度、半年、年度集中召开高层会议，不会天天开会。

在条件允许的情况下，尽可能采用较短的 T 来做决策的好处是可以快速验证、调整和迭代。较短的 T 在决策中意味着在一定的时间内有更多次的试错机会，这对于试探性很强的新兴产业来说更有吸引力。

17.3.3　不要“输不起”

鉴于上面这些原则，决策中涉及的问题就是如何衡量一段时间内一切试错决策中的成功概率、试错的总成本及消耗的时间。

既然是试错，就意味着要把这种投入当成一种必需的成本来看待。所以，既没必要纠结，也一定要留有余地——不要让试错成本高到让自己“输不起”的程度，这也是在做出重大决策时，任何人、任何组织都会非常谨慎的原因。

对于中小规模的公司，尤其是在有了一定健康的流水以后，通常会比较严格地采用预算制进行开支的控制，其目的也是为了保证不要出现这种“输不起”的情况。而一些人才济济、开拓性思维成员较多的公司，通常也会培养一种去中心化的项目激励制度。大致的形式是：公司只建立流量入口性质的平台和服务性质的平台，在一年中把平台相关的预算和服务性质的预算做足，然后让每个人参与其中；或者让一个人或三五成群形成小组织，每个小组织都有进行项目提案和项目组织的权力。当一个项目提案获得批准后，其消耗的资源会从预算池中分期分批扣除。对一个运作成熟的公司来说，这种方式的好处是能够将大决策中心简化为小决策中心，降低单个决策的风险和试错成本，加快决策执行和试错验证的速度。这种平台一旦成熟，其面对的就不是公司内部的个人了，而是面向全体社会，形成全社会层面的去中心化市场决策。

苹果的应用商店是一个非常典型的化简决策模型的平台，每个人都能利用自己在某一方面的认知特长做出一个非常好的应用，在苹果应用商店发布自己的 App 产品。这种应用的边界清晰，迭代短促，决策成本较低，适合这种规模产品的市场角逐。如果苹果不开放 iOS 的开发标准，而是把这些工作统统留在苹果公司内部来做决策——每个月开个会，不同分类 App 的副总裁汇报开发计划，做成本预算，再在这几百个方案中选择最为可靠的十几个方案作为月度开发计划——这种决策本身就要消耗一两个月甚至更久的时间，而且这种封闭性天然拒绝了全球化市场中精英头脑里的创造性思维。所以，相比之下，让全社会参与决策的方式显然更有利于市场的发展。

美国的著名游戏门户平台 Steam[122] 也非常繁荣，在这个平台上每天都有新的游戏发售，也有玩家在平台上对战和交流，更有周边的出版物、硬件、纪念品可以买卖。可以说，Steam 形成了一个非常良性的游戏平台生态。Steam 本身不做游戏开发，只负责提供流量入口和支撑服务，并从中抽取一定量的佣金，各个游戏的“生死”由市场决定。这样的方式解放了游戏开发人员，也解放了 Steam 平台自身，大家都在做自己更擅长的事情，而把和主业无关的东西“外包”给别人——这绝对是重要的决策原则。

17.3.4　留得青山在

任何时候都不要动辄采取“不成功，则成仁”的赌徒心态，尤其不要轻易把自己的生命和名誉投入到一些成功率极为不确定的决策中，因为它们都是不可再生的。换句话说，这种损失是无法用金钱挽回的。所以，不要责怪一些企业的负责人总是对决策问题谋而不决，他们自然有自己的苦衷，而这种“不决”恰恰是一种对企业负责任的表现。

越是对公司的格局和现金流影响重大的决策，就要越谨慎地做出——没办法，这种投入中的 I 足以使一个公司翻盘的“血库”耗尽。所以，公司几乎在任何情况下都相对更欢迎那种 I 和 T 的值都比较小的项目决策。靠大量试错来遴选适应客观需求的项目，不是因为决策者不够聪明，而是因为客观认知太有限。所以，对 ROI 涉及的指标 R、I、T 的评价永远都是片面的，永远都有判断不准的概率存在。这种局限性来自人类认知本身，只能尽量减少，无法完全克服。

不必纠结，我们只需要把握以下原则。

（1）大趋势要正确

从整个人类历史发展进程来看，整体趋势就是那些能改进生产力、发展生产力的科学技术会被传承和发扬。因为这种技术本身就在产生利益，所以必然会受到欢迎。而决策对象是一个改进生产力或发展生产力的项目，所以应该在全社会范围内受到追捧。

中短期的趋势也可以做参考。这种趋势通常是由因各种各样的事件刺激而产生的“市场情

122　官网地址：http://store.steampowered.com/。

绪”造成的。例如，AlphaGO 战胜李世乭之后的围棋热卖，以及一些深度学习领域的科普书籍一度畅销，就是中短期的趋势。至于它们是否一定会形成中长期的趋势则不确定，起码围棋应该不会长时间热卖。而由于深度学习会间接帮助改进和发展生产力，所以从这个角度来说，深度学习领域的科普书籍畅销所形成的长尾或增长趋势要比围棋热卖来得更长久。

（2）以小迭代为主

短期与快速试错，不要随便赌上整个公司的身家性命。

（3）以人为本

不要被理论束缚。人生在世界上就会有各种需求。我的观点是，只要这个需求主张能够与其他方面的需求主张互相满足，就构成了产生最基本的市场交易的要件。只要这种交易中没有故意的欺瞒或者胁迫，能够顺利成交，就基本可以认定为公平。

在交易的过程中，可能会有人认为某种交易的产生并不公平。造成这种情况的原因通常有两种，一是信息不对称，二是自身不得已（客观原因）。所谓“信息不对称”，是指一方由于掌握了更为灵通的市场信息或者对市场变化的把握更为准确，所以可以进行一些看上去“不公平”的低买高卖行为。“自身不得已”的客观原因很多，例如因为急于将一些项目或者产品变现，所以愿意以远低于市场价的价格来换取成交时间的缩短。

不要忘记，对市场这种博弈环境来说，任何条件都是交易的核心要件，缺一不可。一个市场博弈者的选择是自由的，既然选择是基于当时自己认为最好的条件做出的，就别再不断纠结是否公平的问题——公平与否同样是自己强加给自己的（信息不对称；自身不得已）。但如果交易虽然满足了这两者的利益互补关系，却伤害了第三者的利益，那么这种行为在大环境内会受到受损方的压制，其博弈就要看这两种力量的对抗究竟各自胜算几何。还是那句话，“仁者见仁，智者见智”。

17.4　本章小结

数据决策的核心在于对 $ROI=\frac{R}{I\cdot T}\cdot 100\%$ 这个公式的灵活运用。只要运用得当，基本上可以帮助企业在很多问题上快速得到一个足够好的解。而且，该公式的应用极为广泛。

决策对大部分企业而言都是一个艰难的问题。由于太多的问题无法做到有效的量化或量化成本过高，例如名誉、品牌影响力、科技水平等，导致企业还是喜欢用“拍脑袋”的方式做决策。那么，这种量化服务的固定化或者统一化是不是同样能够作为一种产业呢？我们不得而知，但是这件事一定值得人们思考与探索。

在对抗犯罪领域，也可以应用这种模型来进行分析和决策。在制定战略级的法律法规，特别是在制定政策制时，通常都是从宏观上诱导大量的社会人员从事合法劳动，以提高整个社会

的 ROI。在采用战术级的侦查、防范等手段时，通常都是用有限的或尽可能低的成本投入来压制犯罪嫌疑人的 ROI。例如，银行的密码系统会采取多种防范措施（在信道上使用 2048 位的非对称加密），这样即使信道被不法分子监听，其破解的 ROI 将变得极为低下（要花几十万年的时间才能非法获利几百万元，显然不可行）。再如，在治安较差的路段加装监控装置，使犯罪嫌疑人的行踪更容易被发现和监控，从而提高其犯罪成本，降低其犯罪的 ROI。这些都是很好的例证。

第 18 章　案例分析

将数据科学中的思想融会贯通，应用到实际生产当中，是很多大数据架构师、大数据分析师及所有大数据产业的从业者都希望做到的事情。

任何一个案例都有自己的特性，要解决的主要矛盾也不尽相同，而且很多复杂的案例已经能够单独成书，三言两语把一个案例从头到尾分析清楚未免有点痴心妄想。下面我们不妨通过两个有趣的事例来感受数据科学的应用在具体场景中的指导作用。

18.1　K 线图里的秘密

炒过股票或者进行过黄金、外汇交易的读者对 K 线图应该不会陌生。K 线图是一种表示周期内报价波动信息的图形，由于形状像蜡烛，所以也叫“蜡烛图”（如图 18-1 所示）。

图 18-1　K 线图（EUR/USD，5 分钟）

在每一根“蜡烛”上有 4 个关键的数据信息，分别是开盘价、最高价、最低价和收盘价。K 线图表示的周期可以根据观察者的需要进行缩放，由于每一刻的报价数据都有可能变化，所以这种变化在时域 t 上会出现一个拖影。因为这种数据体量太大，通常不太容易进行观察和分析，所以需要在时间这个维度上进行压缩，常见的方式是在一个 Δt 内取开盘价、最高价、最低价和收盘价绘制图形。Δt 根据分析的需要，也有 1 分钟、5 分钟、15 分钟、30 分钟、60 分钟、1 天、1 周、1 个月这样的周期。

以图 18–1 为例，这是一张欧元兑美元的报价 K 线图，每一根“蜡烛”都表示一个 5 分钟时间段内的数据情况。在 K 线图中，通常情况下，绿色代表收盘价比开盘价高的情况，称为“阳柱”；红色代表收盘价比开盘价低的情况，称为“阴柱”；上下两端拉出的细线称为“影线”，分别表示在这个时间单位中价格所达到的最高点和最低点。

18.1.1　什么是市场

“市场”（Market）原本是指交易的场所，现在泛指交易行为所构成的全体要素。

有几个重要的概念与市场相关，第一个就是商品。商品是指交易双方交易的对象，卖出商品的一方叫作“卖家”，买入商品的一方叫作“买家”。按照经济的定义来看，市场经济（又称“自由市场经济”或“自由企业经济”）是一种经济体系，在这种体系下，产品和服务的生产及销售完全由自由市场的自由价格机制引导，而不是像计划经济一般由国家引导。也就是说，交易行为中有供需双方和交易对象，就已经构成了市场。

在股票或外汇市场中，交易双方使用货币或者有价证券与对方的货币或者有价证券进行交换，双方的定价权是自由的。

18.1.2　谁在控制价格

对一宗交易来说，成交价是非常重要的因素。一般我们习惯把商品的成交价称为“价格”，而且只有成交时候的价格才有意义。按照市场经济的规律，价格是由供需双方的矛盾决定的，是一种自由博弈的结果。这对矛盾就是大量的卖家卖出商品换取货币和买家卖出货币换取商品。在这里，货币本身也是商品，在市场供需中它与其他商品的区别仅仅在于它是一般等价物，是一种用来给其他商品标识价格的计量单位。

有一种说法叫作“有价无市”，通常是指按照卖家喊出的价格，没有买家愿意与之成交。形成这个现象的因素很多，但是总体来说可以理解为卖家喊价超过买家心里预期——有商品供应方，有出售价格，但是没有成交。

通常我们谈到的“市场价格”是指宏观性的成交价格，也就是在市场上大量成交的价格。按照刚刚所说的这种供需自由博弈的观点来看，双方的焦点仅在于成交价格，与动机无关。

1. 低价成交原则

在市场博弈中有一个非常重要的原则就是“低价成交原则”，是指在商品的任何因素都完全一样的情况下，人们总是愿意优先购买价格更低的那种商品。

例如，两个商家同时销售 iPhone 7（128GB，黑色，三网通用版），都是国产行货，同一批次，未开封，保修条件相同，摊位相邻……总之，一切因素对消费者来说完全一样。但是，一个商家售价 6400 元，另一个商家售价 6200 元。一般人会怎么选？理论上，要想观察这个过程，应该在一段时间内分别统计两个商家的成交数量，而且要保证这两个商家在来往的顾客中有相同的流量入口和招待气氛。这个过程太理想了，几乎无法实现。不过，如果真的是无差别商品，价格低的会更吸引人，更容易成交——这是由人类本身“趋利避害”的决策原则决定的。

因此，在市场上买卖双方之间产生的对于某种特定商品的成交价，都是由大量供应的同等产品的最低价格决定的。但也有一些个别的成交案例不符合这一原则，例如免费赠送及商家因急于清仓而进行的降价处理等。只要这些行为在整个市场的体量中足够小，那么它们对价格的影响就可以忽略。

2. 价格的波动

在市场上还有一种现象几乎天天都能看到，那就是价格的波动。价格不是一成不变的，很多因素和动机都会引导价格的上涨和下降。

价格的波动从最简化的角度看，还是供需矛盾不均衡的问题。虽然影响价格的因素很多，例如制造成本压力因素、税收政策因素、人力成本因素等，但最后反映到市场上可以用一个抽象的名词来表示，叫作“市场情绪”。市场情绪普遍存在，但通常在投机交易市场中尤为明显。市场情绪带来的是对某一种或者某一类商品的“买入”或“卖出”的态度。下面我们在投机市场和非投机市场各找一个例子来说明。

“郁金香泡沫”又称“郁金香效应”（经济学术语），源自 1636 年发生于荷兰的一个事件。作为人类历史上有记载的最早的投机活动，荷兰的“郁金香泡沫”昭示了此后人类社会的一切投机活动，尤其是金融投机活动中的各种要素和环节——对财富的狂热追求、羊群效应、理性完全丧失、泡沫最终破灭和千百万人倾家荡产。

曾有人考证，郁金香的原产地是中国，据说在很久以前，商人们骑着骆驼通过丝绸之路把郁金香带到了土耳其。根据文献记载，郁金香在 16 世纪中叶从土耳其传入奥地利，然后从奥地利逐步传向西欧。当时，荷兰是世界上屈指可数的强国，因其独特的气候和土壤条件，很快成为郁金香的主要栽培国之一。

1630 年前后，荷兰人培育出了一些新奇的郁金香品种，其颜色和花型都深受人们欢迎。典雅高贵的郁金香新品种很快就风靡欧洲上层社会，在礼服上别一枝郁金香成为最时髦的穿搭方

式，贵夫人在晚礼服上佩戴郁金香珍品来显示自己的地位和身份。王室贵族和达官富豪们趋之若鹜，争相购买最稀有的郁金香品种，特别是在法国盛行的奢侈之风，把郁金香的价格逐渐抬高。1635 年秋季，名贵品种郁金香的价格节节上升。当时在巴黎，一枝最好的郁金香花茎的价格相当于 110 盎司[123] 黄金。

1634 年以后，郁金香的市场需求量逐渐上升。1636 年 10 月之后，不仅珍贵品种的价格被抬高，几乎所有品种的价格都飞速上涨。短短 1 个月内，郁金香的价格被抬高了十几倍甚至几十倍。但是好景不长，"郁金香泡沫"只维持了一个冬天。1637 年开春之前，"郁金香泡沫"破灭了，郁金香市场一片混乱，价格急剧下降。1739 年的数据显示，有些品种的郁金香价格狂跌到最高价位的 0.005%。[124]

"郁金香泡沫"是一个典型的投机市场方面的例子。下面我们来看一个非投机行为的例子。

2009 年 6 月，与强热带风暴"莲花"登陆福建几乎同时，一股谣言突袭福建，随之劲刮的食盐抢购风席卷宁德、福州、南平、莆田、三明等地。6 月 21 日，宁德市霞浦县突传谣言，说盐场受到污染，食盐将要断货并涨价，引发大量市民抢购食盐，部分乡镇断货，个别地方以前售价 1 元/包的食盐被不法商贩炒至 3～5 元/包。当晚，谣言传至宁德市蕉城区和福州市区，引发抢购，部分商店断货。当时，宁德和福州的日常食盐日订货量分别为 44 吨和 77 吨，但 2009 年 6 月 21 日晚至 22 日 10 时，订货量分别猛增至 321 吨和 716 吨。[125]

这种疯狂抢购的行为背后就是一种"买入"的情绪。大部分市民买入食盐是为了囤积和食用，而不是为了低买高卖。虽然这种行为是非投机行为，但它同样能够起到哄抬物价的作用。

3. 货币的价格

一般来说，对完全相同的商品才容易讨论价格，所以对制式商品进行价格比较，其意义通常更为明显。例如，电脑配件或外设（MP4、PAD、录音笔等）、一手汽车，甚至矿泉水、方便面、调味料等，这些产品的描述指标相对固定，数值清晰，所以很容易进行比较，对于描述指标完全一样的产品，只比较价格就可以做出决定。但是，另外一些产品的比较往往比较困难，例如水果、蔬菜、粮食、服装、鞋帽等。在原材料大宗交易的场合，通常会用比较丰富的技术指标进行考量，但是在零售领域，尤其是在老百姓日常接触的这种零售领域，即使开放了技术指标，也未必有人能读懂，更别说根据这些丰富的指标进行比较和判断了。

但是，对股票、货币等在市场上自由交易的有价证券或外汇期货产品来说，比较就容易得多——作为买家，不管是谁出售的股票或欧元，都是毫无差别的股票或欧元，只需要比较开价高低就可以了。

123　1 盎司黄金约为 31.1 克黄金。

124　来自百度百科，有删节。

125　来自百度百科，有删节。

18.1.3　货币价格的形成

某种商品在一个封闭市场中的价格由卖方和买方共同决定，货币价格的形成也不例外。

外汇市场通常不容易出现所谓的“庄家”操盘，也不容易出现个人或者组织有足够的力量把大量资金投入外汇市场进行恶意投机的情况，原因在于，货币的体量已经大到无法由个人或组织来操控的地步——绝大多数情况下确实是这样。

我们所说的“货币”是指在国际流通市场上普遍受到承认的大币种，例如欧元（EUR）、美元（USD）、英镑（GBP）、瑞士法郎（CHF）、加元（CAD）、日元（JPY）、新西兰元（NZD）、澳元（AUD）等。这些货币分别由各个经济共同体或者国家的央行发行，而且这些组织或国家的央行相对经济体量比较大，在世界经济格局中扮演着重要的角色，所以它们发行的货币有很高的信誉。而另外一些国家的货币，尤其是小币种，例如越南盾、泰铢、菲律宾比索、津巴布韦元（如图 18-2 所示）等，其发行方都是经济体量较小、经济格局较脆弱的国家。因为这些货币出现价格剧烈波动的概率很高，所以在国际市场上通常不会有人愿意大量持有。

图 18-2　10 万亿面值的津巴布韦元

现在的货币几乎不采用金本位[126]制度，货币的报价一般都与美元挂钩。例如，欧元兑换美元为 1.1260，就是说 1 欧元可以兑换 1.1260 美元；美元兑日元为 100.47，就是说 1 美元可以兑换 100.47 日元。这些货币在国际市场上会由于各种原因进行汇兑，大宗的汇兑通常是由国际间结算引起的，而买卖货币的个体可能是某个银行、某个企业，也可能是某个个人，当然还有可能是某个国家的央行。这些参与方都在有形或无形地控制着货币价格的涨跌。

与股票这样的证券一样，货币的报价也是由市场供需决定的，而不是由一个组织就能确定下来的——哪怕它是央行。央行如果想控制货币的价格，也无法直接实现，只能利用各种手段来影响市场的供需，再由市场的供需反映到货币报价中去。

我们来看一个模型。假如现在银行之间要进行欧元买卖交易，也就是一方用手里的美元买欧元，另一方用手里的欧元买美元。既然市场是自由的，那么所有想要卖出欧元买入美元或者卖出美元买入欧元的人，都可以参与报价和交易。

以美元作为计价单位，会形成一张表（如表 18-1 所示）。表中显示了市场上目前可以交易的欧元数量，前 4 行是卖家的数据，后 4 行是买家的数据，所有信息一目了然。例如，卖家 A 希望以 1.1200 美元的价格卖出 500 欧元，买家 E 希望以 1.1195 美元的价格买入 2500 欧元。这

126　“金本位”即“金本位制”（Gold standard）。金本位制就是以黄金为本位币的货币制度。在金本位制下，每单位的货币价值等同于相应重量的黄金（即“货币含金量”）。

种情况实际上是不存在成交的，因为其中没有合适的成交价。如果你是一个欧元的卖家，且希望立刻成交，最理想的情况是以 1.1195 美元的价格卖出 2500 欧元，即与买家 E 成交；而如果你是一个欧元的买家，且希望立刻成交，最理想的情况是以 1.1197 美元的价格买入 1 欧元，最多买 200 欧元。

表 18–1　欧元价格表

买/卖家	方　向	报　价	数　量
A	卖出	1.1200	500
B	卖出	1.1199	500
C	卖出	1.1198	1000
D	卖出	1.1197	200
E	买入	1.1195	2500
F	买入	1.1194	1000
G	买入	1.1193	3000
H	买入	1.1192	2000

在股票和其他期货市场中，这个表的信息是完全透明的。但是，在外汇散户的交易环境中看不到这样一个表，只能通过 K 线图上的成交价来看报价。仍以表 18–1 中的报价为例，如果目前市场上买入欧元的量比较大，那么在以 1.1197 美元的价格买入 200 欧元后，会再以 1.1198 美元的价格成交。当报价 1.1198 美元的 1000 欧元被卖出后，再想买入欧元，只能以 1.1199 美元的价格成交。只要买入的量足够大，价格会一直上涨，我们看到的 EUR/USD 的 K 线图就会呈现上涨的趋势；相反，如果卖出的量足够大，成交价格会不断下跌，EUR/USD 的 K 线图就会呈现下跌的趋势。从表 18–1 反映的情况看，似乎价格更容易向上移动，因为同样是交易 2000 欧元，买入价已经提升到了 1.1200 美元，卖出价则会维持在 1.1195 美元，这种情况就是“买盘压力”较大的情况。

补充两点。第一，实际上这张表不会一直老老实实地待在那里，而是会不断变化，因为每时每刻都有人参与到市场博弈当中。第二，外汇交易量是非常巨大的，银行之间的交易量通常都是数以百万计，这张表只是为了便于理解模型而做了简化。

18.1.4　零和博弈

不论是股票交易还是外汇交易，都有一个和亚当 · 斯密[127]的《国富论》中的财富创造理论相悖的特点。亚当 · 斯密在《国富论》中提到，财富的积累是通过分工来增长的，不同国家、不同地区、不同人所擅长的生产领域不同，应该让擅长从事某一领域生产的人专职生产某一种产

127　亚当 · 斯密（1723 年 6 月 ~ 1790 年 7 月），英国人，经济学的主要创立者。

品，这样对于任何一个生产个体来说都达到了效率最大化。在世界范围内，相同时间生产的产品总量就会增加，产品的增加也就意味着财富的增加，以及每个人被分到的产品和财富数额的增加。这也就意味着，在商品交换的过程中，这种彼此生产的产品被市场所认可，接下来会进一步刺激生产者对生产技术的改进，并进一步扩大这种产品的生产能力。

但是，在股票和外汇交易的过程中是看不到这样一种现象的。尤其是股票交易的过程中不产生财富，也就看不到财富的增加。一个人在市场中赢取的财富是另一个人所失去的财富，这也叫“零和博弈”。一种典型的零和博弈就是赌博。在赌场里，每个参与赌博的人在赌桌上赢取的筹码是其他人输掉的筹码，在赌场这个封闭的环境中，财富（筹码）总量是不会增加的。

在股票和外汇交易中，我们仔细想想也能知道这是一种“对赌”行为。在整个市场环境中，对于任何一种股票或任意一组货币对的供需矛盾来说，“有买必有卖，买卖必相等”是一个根本的原则。任何一笔买多成交的时候，都有一笔对应的沽空[128]成交。同样，任何一笔沽空成交的时候，必然有对应的卖空成交。我们在 18.1.3 节的模型中已经看到这种现象了。

到目前为止，笔者都从来没有认为股票交易和外汇交易是一种对社会经济和生产力发展有明显作用的行为。在这些市场中充斥着大量的投机交易，这些交易中出现的“低买高卖”和“高卖低买”的行为，与黄牛党倒卖车票、门票及赌博的行为类似，都是通过市场供需的不平衡和信息的不对称来盈利的行为，即使在经济制度完善和科技发达的今天，一个国家都不可能靠这类行为实现振兴，而是要提高国家的国际地位和综合竞争力。所以，对这类事情，我个人持谨慎态度——它是市场经济发展到一定程度的产物，在对其本质和规律进行充分了解的情况下，可以适当尝试参与这种市场博弈过程。在你确认它是一种可预测的行为并可以为你稳定盈利之前，请不要过于沉迷。

18.1.5 涨跌都盈利

在股票和外汇的交易体系中，有一种涨跌都可以盈利的现象。这对于在传统的“单边”交易中只认为低买高卖能够盈利的人来说，需要在脑子里转一个弯才能想明白。

例如，现在我持有 100 美元，当前美元兑日元汇率（USD/JPY）是 100.00，也就是说，100 美元可以兑换 10000 日元。此时，我判断美元兑日元汇率会下跌，也就是日元会升值，于是用 100 美元换成 10000 日元。一段时间后，美元兑日元汇率下跌到 95.00，此时我持有的 10000 日元可以兑换成 105.26 美元。于是，我再将日元兑换成美元，在这个过程中获得了 5.26 美元的收益。

还是以刚刚这个场景为例。现在我持有 100 美元，当前的美元兑日元汇率（USD/JPY）是 100.00。此时，我判断美元兑日元会上涨，也就是美元会升值，日元会贬值。于是，我用手中的美元做抵押，借入 10000 日元，在市场上抛售，换取 100 美元，这时我就持有 200 美元了。当

128 卖出。

美元兑日元汇率上涨到 105.00 的时候，用其中的 95.24 美元就能换回 10000 日元。我将其兑换，将借款归还贷款人，在这个过程中获得了 4.76 美元的收益[129]。

这样一个过程就是“涨跌都盈利”的过程。由于外汇波动的幅度非常有限，像刚刚这种 USD/JPY 从 100.00 上涨到 105.00 的 500 个点[130]的波动，通常最快也要经过几个月才能达到，而一般一天内的波动仅有几十个点而已。所以，在外汇交易中通常会有很多市商提供很大的杠杆来进行“波幅放大”，常见的有 1∶100、1∶200、1∶500 等尺寸的杠杆。有些人往往会有误解，认为杠杆越大，风险越高。不过，有外汇交易或者带有杠杆配资的股市交易经验的人会很清楚，在其他条件不变的情况下，风险的大小只与持仓的头寸[131]大小有关。

18.1.6　价格的预测

在外汇体系中，价格的预测是很多操盘手都非常希望掌握的技巧。从大的分析方法来看有两种，一种是“基本面”分析，另一种是“技术面”分析。这两种方法各有优缺点，我们来简单了解一下它们的分析思路。

所谓“基本面”分析是指从宏观层面判断货币的供需，进而在一个中长期的时间段里判断货币的价格走向，这种方法通常不适合做短期判断。例如，2014 年 9 月 4 日晚，欧洲央行意外降息，并预告同年 10 月将推出量化宽松政策。这种从央行层面释放出来的信号，会影响大量的欧元供应，使其流向市场，所以欧元兑换主流货币的幅度会在接下来的几个月内大幅降低。直到 2015 年 3 月 31 日，EUR/USD 从 2014 年 9 月的 1.3100 附近下跌到 1.0500 附近，下跌了约 2600 点（如图 18–3 所示）。

图 18–3　EUR/USD（月 K 线）

所谓“技术面”分析是指从外汇报价指标及“形态”上发现其波动规律并进行相应的市场迎合。

对于“基本面”和“技术面”两种方法，不同的分析流派都有自己的一套分析体系。“基本面”分析的依据主要是宏观数据、经济指标、汇市经验及货币供应模型的拟合。“技术面”的分

129　这个过程中可能会产生一定数量的隔夜利息

130　USD/JPY 这种货币对，0.01 是一个点单位。

131　在外币交易中，“建立头寸”是开盘的意思。开盘也叫“敞口”，就是买进一种货币，同时卖出另一种货币的行为。

析流派比较多，有“艾略特波浪理论”[132]、“江恩理论”[133]、“道氏理论”[134]等。“技术面”分析有一个特点，即相对来说不太注重基本面的趋势。“技术面”分析人员认为，不论是什么行为对市场造成的影响，最后都会演变成买多或沽空的市场情绪，而且会在 K 线上体现出来。

此外，有人将“趋势型”和“技术型”两种技巧相结合，在很多场合的效果也比较好。例如，在长期的涨势中看到回调，在回调受阻时继续买多，在上涨遇到阻力时抛出，在大趋势的掩护下这种手段的获利能力通常比较强。在很多股市和汇市的网络论坛里，我们会听到两种很强的呼声。一部分人赞颂这些理论的高深、精美、玄妙，认为一些人不能掌握这些理论是因为不能理解其高深之处；另一部分人对这些理论进行非常尖锐的批驳，往往用最尖刻的话语来批判这些理论是骗人的，人们无法靠这些理论发财致富，真正发财的是靠鼓吹这些理论去卖书、卖课程的人。问题就摆在这里，究竟谁对谁错？这个场景在本书的前面已经有过类似的假定，我坚信这是因为不同的人在这个“局”中有着不同的利益取舍，但我同样不能排除这种可能——前者是卖教程的，后者是因为没学会所以跳出来大骂理论的错误。不管这两类人究竟出于什么动机而发出这样的声音，作为一名数据科学的信徒，我们应该把握科学的底线与原则。

（1）原则一：复杂到谁都理解不了的东西不能被认定为科学。只有极少数人在极少数的场景下能够应验的东西，本身就违背了科学性。

（2）原则二：科学的东西都具有体系性、精确性。如果其中之一不具备，那它就不是科学的，不论是谁在为它背书，都不能被认为是科学的，因为它不符合科学的基本特征和定义。

18.1.7 形态

在价格预测中有一个相对比较简单的工具为大多数股市和汇市爱好者喜欢，那就是“形态”。“形态”这东西有点像围棋里的“定势”。不论是在“江恩理论”还是在“道氏理论”中，都有关于“形态判定对价格走势有影响”的论述。我们选取其中几个例子来做一下说明。

1. 头肩顶

如图 18-4 所示，这个形态就像一个人的头和肩膀一样，所以命名为“头肩顶”。这是一种比较典型的反转状态，一个 K 线的走势走到一定的高度后上冲乏力，无法突破并继续向上，转而向下，是入场的好时机，做空。

---

132 “艾略特波浪理论”由美国证券分析家拉尔夫·纳尔逊·艾略特（IiR.N.Elliott）提出，用于分析股市走势。由于其“波浪”判断的滞后性，只有当“波浪”图形完整呈现时才有对应解释，所以为许多业界人士诟病。

133 “江恩理论”是投资大师威廉·江恩（Willian D.Gann）通过对数学、几何学、宗教、天文学的综合运用建立的独特分析方法和测市理论，是他结合自己在股票和期货市场上的骄人成绩和宝贵经验提出的，包括江恩时间法则、江恩价格法则和江恩线等。

134 “道氏理论”创始人查尔斯·道声称，其理论并不是用于预测股市，甚至不能用于指导投资者，而是一种反映市场总体趋势的晴雨表。大多数人将道氏理论当成一种技术分析手段——这是一种让人非常遗憾的观点。

2. 三重顶

如图 18-5 所示，三重顶形态是一种 3 次尝试上冲都没有突破的形态，出现了 3 个顶点。这也是一个比较典型的入场时机，做空。

图 18-4　头肩顶

图 18-5　三重顶

3. 双重顶

如图 18-6 所示，双重顶也叫“M 顶”，是一个两次尝试上冲都没有突破的形态。这也是一个比较典型的做空时机。

图 18-6　双重顶

头肩顶、三重顶、双重顶都是由市场多空力量争夺产生的翻转形态，一般认为空头明确的强度为：头肩顶 > 三重顶 > 双重顶。三者也分别有对应的对称形态。例如，将如图 18-6 所示的图形沿横轴翻转 180 度，翻转后的图形，其含义同样是没有突破，只不过是没有向下突破成功，属于典型的做多时机。

4. 收敛三角形

在 K 线图震荡中，可能会出现如图 18-7 所示的这种三角形振幅收拢的图形，这也被认为是一种多空力量争夺的形态。当一侧出现突破后，通常会走出至少 1 个颈线的高度（颈线高度就是左侧三角形开口的竖直高度）。

收敛三角形不仅可以呈现向上突破的状态，也可能呈现向下突破的状态。

5. 扩散三角形

扩散三角形和收敛三角形的开口方向相反，在压力线（虚线）部分突破以后就会向一侧走出至少 1 个颈线的高度，上下两个方向都有可能突破，如图 18-8 所示。

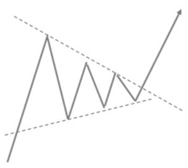


图 18-7　收敛三角形

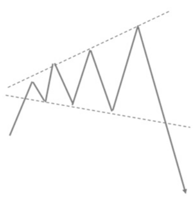


图 18-8　扩散三角形

6. 箱体震荡

如图 18–9 所示的 3 种形态都叫作“箱体震荡”，这种多空力量的角逐就像是在一个通道里进行，无法突破，直到某一瞬间突然突破后，会向突破方向至少涨（跌）1 个颈线的高度。

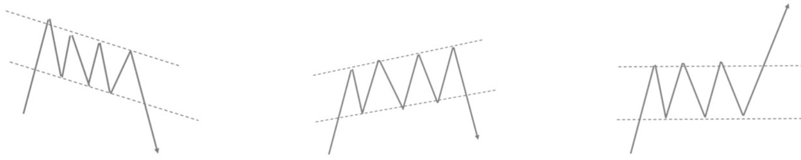


图 18–9　箱体震荡

关于 K 线图的形态，在其他资料上会看到更多的总结，这里列出的并不全面。现在普遍认为，形态是一种对市场上产品多空情绪的直接表达，在反复的多空情绪博弈中突然产生了一种类似市场“一致认为”或者“共振”的现象，从而导致突破或向一个方向走出比较远的情形。

关于形态与走向和幅度的关系，如果读者感兴趣，可以用统计的方法对历史数据进行分析，看看某种形态出现之后发生价格波动的概率究竟几何。关于形态与价格波动概率的实验，在本书中就不讨论了。

18.1.8 K 线图周期

作为一种研究对象，有很多人热衷收集 K 线图的历史数据并进行分析。现在市面上能够找到的性价比比较高的 K 线图数据大都是基于 MT4 平台的数据，是售卖数据的商家从 MT4 平台的历史数据中导出的，最小分度为 1 分钟，也就是说，会给出每分钟的开盘价、最高价、最低价和收盘价。

以分钟为单位的数据可以直接导入并生成 m1 图（1 分钟单位）。以 m1 图为基础，经过合并，还可以生成 m5 图（5 分钟单位）、m15 图（15 分钟单位）、m30 图（30 分钟单位）、h1 图（1 小时单位）、h2 图（2 小时单位）、h4 图（4 小时单位）、d1 图（1 天单位）、w1 图（1 周单位）、mon1 图（1 月单位）等。从理论上讲，还可以生成更长时间单位的图，只不过在分析层面的应用率太低，所以 mon1 这个粒度一般就够用了。

不同的操盘手和分析人员所习惯的观察周期是不一样的，这直接导致他们习惯分析的图也不一样。喜欢做日内单的操盘手通常喜欢看短周期的图，如 m1 图、m5 图、m15 图等；而喜欢做中长期单的操盘手则喜欢看 m30 图、h1 图、h2 图或更长周期的图。

观察不同的周期单位图我们可以发现，它们就像相似型一样，会出现不同周期的各种相近形态。要注意的是，各种周期单位图都有人在研究，在对任何一个周期或者几个周期结合分析时，都会尝试分析或者套用 K 线图形态判断的理论，并将其作为入场和离场的依据。下面我们将要讨论的就是如何用统计认知的方法量化入场和离场的成功概率及获利量大小的问题。

18.1.9　造市商与点差

在正式开始分析之前，我们还要补充一个知识点——造市商。造市商（Market Maker）在这里泛指所有为在汇市中交易的人提供报价和交易接口的商人。在具体交易细节上，这些商人彼此之间还有区别，有 MM、ECN、STP 等多种模式。这些在具体交易的时候再去选择吧，现在我们介绍一下造市商要做哪些事情。

理论上讲，一个造市商从拥有大交易流量的银行实时拿到报价，并对外提供报价和交易接口。如果忽略银行这个环节，我们能看到什么呢？例如，一个商人向我们报价，当前 EUR/USD 的价格为 ask 1.1260/bid 1.1260，有一个 0.0002 的差异，这个差异叫作“点差”。点差相当于造市商与交易人进行交易的差价，表示在这一刻将以 1.1260 的价格购买 EUR/USD 这个货币对，并以 1.1262 的价格卖出这个货币对，即在这一刻做多 EUR/USD 时会以 1.1262 成交，而做空 EUR/USD 时会以 1.1260 成交——与造市商的报价相反。这一点不难理解，就像粮站收粮一样，1 元 1 斤买入，1.1 元 1 斤卖出，赚取差价，绝对不可能是反过来的情况。

不同的造市商提供的点差是不一样的，而且在不同时刻提供的同一种货币对的点差也是不一样的。在一般情况下，点差都比较低或稳定，而在大行情出现的情况下，点差也会有比较大的波动，有时甚至会有几十个点差的情况。

点差的存在给造市商的交易提供了利润。当有人交易 1 手[135] EUR/USD 的时候，造市商将在交易人平仓时获得 20 美元的利润。交易的量越大，造市商所获得的利润就越高。这种利润对交易人来说是一种成本，通常点差越小，对交易人越有利。

造市商理论上应该把交易人的头寸直接拿到银行去交易，但是由于各方面的原因，他们有时也会自己持有一些和交易人相反的头寸，这种现象就是俗称的造市商与交易人“对赌”。这种造市商会给人非常不安全的感觉，因为交易人赚到的钱就是造市商亏损的钱——如果我们被置于只有自己亏损造市商才会赚钱的境地，后果通常不会太好。在这里我们只讨论这种点差对盈利分析的影响，如果真的想做交易，建议还是选择体量大、监管正规的造市商平台开户。

18.1.10　科学分析

1. 导入数据

如果你会使用 MT4 平台的 MQL 语言做开发，就可以将 MQL 作为分析工具。MQL 是 MT4 平台的专用语言工具，对于生成 K 线图上的各种参考指标，提供了非常多的 method。MT4 平台使用的自动交易脚本工具 EA（Expert Advisor）就是用 MQL 语言实现的。在这里，我们使用 MySQL 作为数据载体，使用 Python 作为分析语言进行数据处理。

135　价值 10 万美元的量，写作“1 lot”。

首先，在 CentOS 平台上安装 MySQL 和 Python，然后挑选一种货币对作为分析对象。从网上可以买到一些 1M 粒度的外汇交易历史数据，不过数据质量通常一般，中间会有一些丢失的数据点，所以只要在确认缺失比例不太高的情况下就可以使用。

接着，把数据导入 MySQL 数据库。以我们研究的 EUR/USD 数据为例，其格式如图 18-10 所示。其中，“TICKER”字段为货币对名称，“DTYYYYMMDD”字段为日期，“TIME”字段为时间，“OPEN”字段为开盘价，“HIGH”字段为高点价，“LOW”字段为低点价，“CLOSE”字段为收盘价，“VOL”字段为成交量（在这个文件里没有意义）。

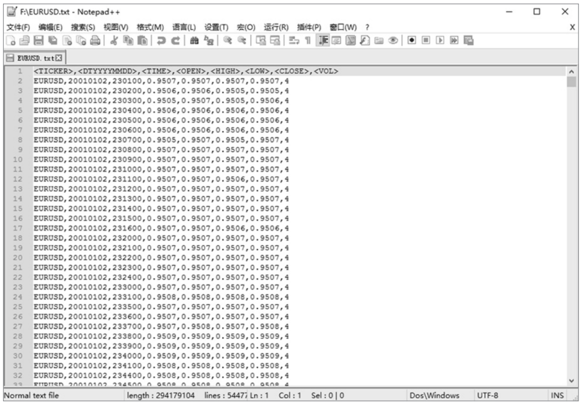


图 18-10 EUR/USD 历史数据

下面我们就编写一段 Python 代码，将这些输入导入数据库。

建立数据库。

```
mysql> CREATE DATABASE FOREX;
```

建立数据表。

```
mysql> USE FOREX;
mysql> CREATE TABLE EURUSD_1M(
DT DATETIME,
OPEN DECIMAL(10,4),
HIGH DECIMAL(10,4),
LOW DECIMAL(10,4),
CLOSE DECIMAL(10,4));
```

编写 Python 代码并导入数据，编辑 insert.py 文件。

```
#!/usr/bin/python
import os,sys,MySQLdb
```

```
try:
  db = MySQLdb.connect(host = 'localhost', user='root', passwd = '111111', db = 'FOREX')

  cursor = db.cursor()
  counter = 0
  input_file = open('/ml/EURUSD.txt','r')

  cursor.execute('USE FOREX;')

  for line in input_file:
    counter =  counter + 1
    date = line.split(',')[1]
    time = line.split(',')[2]
    open_value = line.split(',')[3]
    high_value = line.split(',')[4]
    low_value = line.split(',')[5]
    close_value = line.split(',')[6]
    if date != '<DTYYYYMMDD>':
      datetime = date[0:4] + '-' + date[4:6] + '-' + date[6:8] + ' ' + time[0:2] + ':' + time[2:4]
+ ':' + time[4:6]
      sql = 'INSERT INTO EURUSD_1M VALUES(' + '"' + datetime + '"' + ',' + open_value + ',' + high_value
+ ',' + low_value + ',' + close_value + ');'
      cursor.execute(sql)
    if counter%2000==0:
      db.commit()
  db.commit()
except MySQLdb.Error,e:
  print "Error %s" % (e.args[0])
  exit(1)
cursor.close()
db.close()
```

调用 insert.py 文件。

```
python insert.py
```

这个部分需要多花一些时间。在这个文件中有超过 544 万条数据，建议将 MySQL 调整到 AUTOCOMMIT=0 的状态，然后每 2000 条提交一次。

```
wc -l /root/Desktop/EURUSD.txt
5447761 /root/Desktop/EURUSD.txt
```

2. 观察数据

数据完全导入后，可以用聚合函数来观察，看看这些数据在一天内的波动是如何分布的。使用 vim 命令编辑 query.py 文件。

```
vim query.py
```

编写 Python 代码，对每天的振幅进行统计。

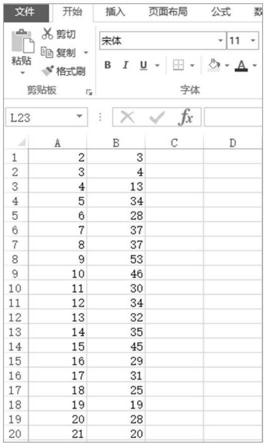
```
#!/usr/bin/python
import os,sys,MySQLdb

try:
  db = MySQLdb.connect(host = 'localhost', user='root', passwd = '111111', db = 'FOREX')

  cursor = db.cursor()
  #result_set=cursor.fetchall()
  #print result_set
  counter = 0
  input_file = open('/ml/EURUSD.txt','r')

  cursor.execute('USE FOREX;')
  sql = 'SELECT PIPS, COUNT(*) DAYS FROM   (SELECT      DATE_FORMAT(DT,"%Y-%m-%d") AS THE_DATE,
(MAX(HIGH)-MIN(LOW))*10000 AS PIPS FROM EURUSD_1M GROUP BY DATE_FORMAT(DT,"%Y-%m-%d")) AS TEMP
GROUP BY PIPS;';
  cursor.execute(sql);
  result = cursor.fetchall()
  for row in result:
    print str(row[0]) + ',' + str(row[1])
except MySQLdb.Error,e:
  print "Error %s" % (e.args[0])
  exit(1)
cursor.close()
db.close()
```

调用 query.py 文件。

```
python query.py > amp_group.csv
```

下载该文件并使用 Excel 打开。如图 18–11 所示，在这个 Excel 文件中，A 列是每天的波动幅度（单位是“点”，即“pip”），B 列是在这 4848 个交易日中有多少天有对应的波动幅度。

如果感觉这样不够直观，那就画幅柱状图来看看。如图 18–12 所示，横坐标是波动的点数，纵坐标是波动点数对应的天数。这个图形中间凸起的部分看上去有点像正态分布，中点在 80 左右（平均值不是 80，实际的平均值比 80 要大），极高波幅日的出现是小概率事件，绝大部分交易日的波幅都在 225 个点以内。大部分日子的波幅比较小，就是左侧的这个高点，中点在 9 附近。



	A	B	C	D
1	2	3		
2	3	4		
3	4	13		
4	5	34		
5	6	28		
6	7	37		
7	8	37		
8	9	53		
9	10	46		
10	11	30		
11	12	34		
12	13	32		
13	14	35		
14	15	45		
15	16	29		
16	17	31		
17	18	25		
18	19	19		
19	20	28		
20	21	20		

图 18–11　波幅统计表

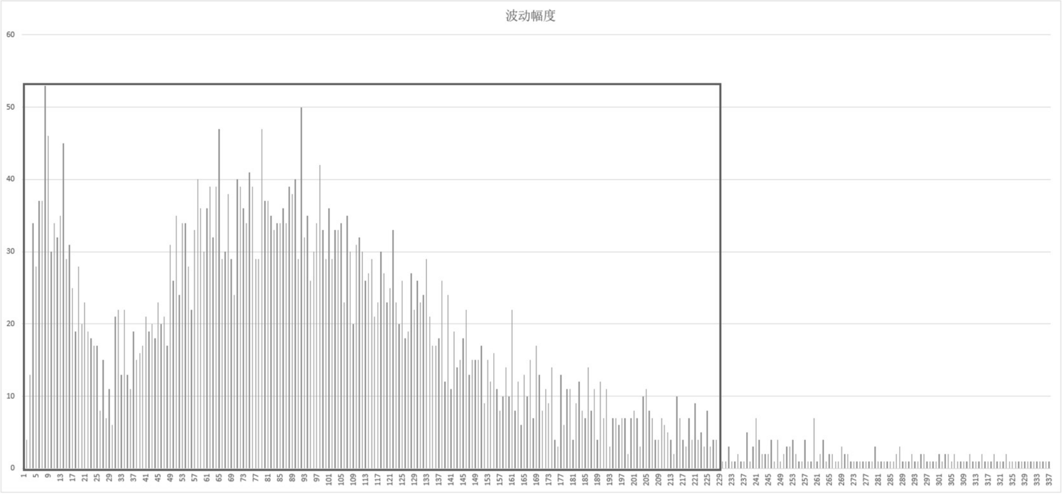


图 18-12　波幅统计图

下面我们分析这些波幅值时，就可以考虑围绕这些已经考察明确的数据来设置范围参数了。在一个点差为 2 的平台上，如果我们只能捕捉到极低波幅的涨势/跌势，例如只有 4 个点，那么下单和平仓的正确率要接近 100% 才行。但是，如果我们能捕捉到波幅在 10 个点以上的情况，就允许存在一定概率的错误。在这里，由于没有具体的策略，所以无法得出这个概率是多少。盈利大小是指每次下单时的点和平仓时的点的差值与方向的乘积的总和。

3. 统计分析

如果我们手里只有这些 K 线图的 m1 数据，并坚信一个涨势/跌势在此前市场酝酿情绪的阶段能够看出端倪，就可以尝试把这个过程描述为一个由前面的数据影响后面数据的非线性的预测过程。

这里对数据进行统计分析和建模的过程，实际上和其他数据分析过程没有本质区别，都是遵循这样的步骤：观察数据→建立假说→训练模型→验证模型→调整模型。

如果建立的假说是刚刚这种前面的数据影响后面的数据的时序相关模型，可以写作：

$$y = f(x_{j-n..}x_j)$$

其中，j 表示当前的时间节点，n 表示一个超参数——表示应该向前看多远的一个滑动窗口的大小。n 的值可以人为调整。

对于涨幅和跌幅的大小，可以尝试用回归模型来做，也可以考虑用分类模型来做。这两种方式的精度要求不太一样。回归模型期望得到一个具体的幅度值（多少点），而分类模型期望得到一个涨幅的分类（暴涨、大涨、小涨、持平、小跌、大跌、暴跌，以及更粗略或者更细致的分类值）。

如果是回归模型，就需要使用线性回归、逻辑回归等进行建模、假设并训练。如果是分类模型，就需要使用决策树、随机森林、SVM 等进行建模拟合。如果使用线性回归、决策树等逻辑上相对简单、有明确参数计算解释的模型进行拟合，在将来把这些条件转化为 MT4 平台上可以使用的 MQL 语言时就会比较方便；如果想使用神经网络进行学习，除了要注意过拟合这种问题以外，还要注意策略很可能无法方便地转化成 MQL 语言，在这种情况下，恐怕除了把这个模型当成一个“黑盒子”一样的“外挂”以外别无他法。

在这里我们只尝试使用一种模型进行分析，主要目的是演示分析和建模的过程，以及给出评价模型质量的思路。其他的方式希望读者能触类旁通，自己尝试。

建立滑动窗口

建立一个滑动窗口，用这个窗口来捕捉那些在一定时间内涨幅或跌幅达到一定幅值的时间段。在这个环节，首先可以尝试找出在 1 小时内涨幅超过 10 点的“小涨”，涨幅超过 20 点的“大涨”，以及涨幅超过 40 点的“暴涨”，看看所有的数据中有多少能够满足这样的条件。重复一下：这种假说只是一种假设，肯定还有其他分类方法。

在字段 DT 上添加一个索引，以加快检索速度。

```
ALTER TABLE EURUSD_1M ADD INDEX IDX_DT (DT);
```

编写 Python 代码，实现滑动窗口。

```
#!/usr/bin/python
import os,sys,MySQLdb

try:
  db = MySQLdb.connect(host = 'localhost', user='root', passwd = '111111', db = 'FOREX')

  cursor = db.cursor()
  input_file = open('/ml/EURUSD.txt','r')

  cursor.execute('USE FOREX;')
  sql = 'SELECT * FROM EURUSD_1M'
  cursor.execute(sql);
  result = cursor.fetchall()
  for i in range(0, cursor.rowcount):
    startdt = str(result[i][0])
    startpip = str(result[i][4])
    cursor1 = db.cursor()
    cursor1.execute('USE FOREX;')
    sql1 = 'SELECT MAX(HIGH)-' + startpip + ' AS RISE_PIP, ' + startpip + '-MIN(LOW) AS FALL_PIP, 
CASE WHEN '+ startpip + '<>MIN(LOW) THEN (MAX(HIGH)-' + startpip + ')/(' + startpip + '-MIN(LOW)) 
ELSE (MAX(HIGH)-' + startpip + ')/0.0001 END FROM EURUSD_1M WHERE DT BETWEEN "' + startdt + '"
```

```
AND DATE_ADD("' + startdt + '", INTERVAL 60 MINUTE)'
    cursor1.execute(sql1)
    result1 = cursor1.fetchall()
    if result1[0][0]>=0.0040:
      output = 'RISE3'
    elif result1[0][0]>=0.0020:
      output = 'RISE2'
    elif result1[0][0]>=0.0010:
      output = 'RISE1'
    elif result1[0][1]>-0.0010:
      output = 'NONE'
    elif result1[0][1]<=-0.0010:
      output = 'FALL1'
    elif result1[0][1]<=-0.0020:
      output = 'FALL2'
    elif result1[0][1]<=-0.0040:
      output = 'FALL3'

    output = startdt + ',' + str(result1[0][0]) + ',' + str(result1[0][1]) + ',' + output + ','
+ str(result1[0][2])
    print output
    cursor1.close()
except MySQLdb.Error,e:
  print "Error %s" % (str(e.args[0])+':'+str(e.args[1]))
  exit(1)
cursor.close()
db.close()
```

这个滑动窗口实现的是：从数据中的第 1 分钟开始滑动，窗口大小为 60 分钟，每次滑动 1 分钟，每次滑动都要看向这一分钟后窗口内的波动情况，查看其中的 HIGH 和 LOW 与当前收盘价 CLOSE 的关系。这种关系大致上分 3 种情况。

第 1 种情况是 CLOSE 在 HIGH 与 LOW 之下，这是一种比较明显的涨势，如图 18-13 所示。

第 2 种情况是 CLOSE 在 HIGH 与 LOW 之间，这是一种“震荡”的态势，如图 18-14 所示。

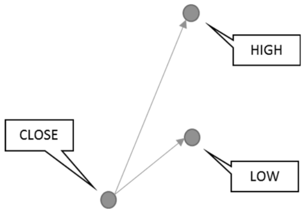


图 18-13　CLOSE 在 HIGH 与 LOW 之下

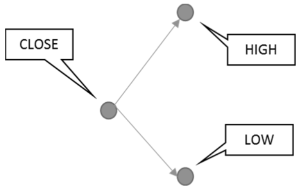


图 18-14　CLOSE 在 HIGH 与 LOW 之间

第 3 种情况是 CLOSE 在 HIGH 与 LOW 之上，这是一种“下跌”的态势，如图 18-15 所示。

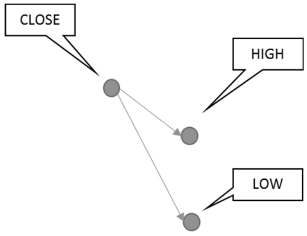


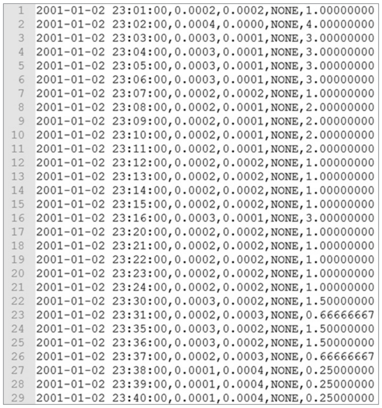
图 18-15　CLOSE 在 HIGH 与 LOW 之上

通过滑动窗口把这 3 种情况在每一分钟“向后看”的过程中都找出来。在这里，为了给出一个相对清晰的分类，把态势分成了 7 种，分别是涨幅超过 40 个点（RISE3）、涨幅超过 20 个点（RISE2）、涨幅超过 10 个点（RISE1）、涨跌幅度在 10 个点以内（NONE）、跌幅超过 10 个点（FALL1）、跌幅超过 20 个点（FALL2）和跌幅超过 40 个点（FALL3）。

调用这个 Python 文件，把内容输出到一个名为 slidewindow.csv 的文件中去。

```
python slidewindow.py > slidewindow.csv
```

下载文件，如图 18-16 所示。



```
2001-01-02 23:01:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:02:00,0.0004,0.0000,NONE,4.00000000
2001-01-02 23:03:00,0.0003,0.0001,NONE,3.00000000
2001-01-02 23:04:00,0.0003,0.0001,NONE,3.00000000
2001-01-02 23:05:00,0.0003,0.0001,NONE,3.00000000
2001-01-02 23:06:00,0.0003,0.0001,NONE,3.00000000
2001-01-02 23:07:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:08:00,0.0002,0.0001,NONE,2.00000000
2001-01-02 23:09:00,0.0002,0.0001,NONE,2.00000000
2001-01-02 23:10:00,0.0002,0.0001,NONE,2.00000000
2001-01-02 23:11:00,0.0002,0.0001,NONE,2.00000000
2001-01-02 23:12:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:13:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:14:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:15:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:16:00,0.0003,0.0001,NONE,3.00000000
2001-01-02 23:20:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:21:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:22:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:23:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:24:00,0.0002,0.0002,NONE,1.00000000
2001-01-02 23:30:00,0.0003,0.0002,NONE,1.50000000
2001-01-02 23:31:00,0.0002,0.0003,NONE,0.66666667
2001-01-02 23:35:00,0.0003,0.0002,NONE,1.50000000
2001-01-02 23:36:00,0.0003,0.0002,NONE,1.50000000
2001-01-02 23:37:00,0.0002,0.0003,NONE,0.66666667
2001-01-02 23:38:00,0.0001,0.0004,NONE,0.25000000
2001-01-02 23:39:00,0.0001,0.0004,NONE,0.25000000
2001-01-02 23:40:00,0.0001,0.0004,NONE,0.25000000
```

图 18-16　slidewindow.csv 文件内容

第 1 个字段是当前时间；第 2 个字段是未来 60 分钟内的高点减去当前这一分钟收盘价的值，也就是涨幅；第 3 个字段是当前这一分钟的收盘价的值减去未来 60 分钟内的低点值；第 4 个字段是第 2 个字段值和第 3 个字段值的商，这个值可以粗略地分为正值较大（如图 18-17 所示）、正值较小（如图 18-18 所示）、负值较大（如图 18-19 所示）和负值较小（如图 18-20 所示）4 种情况。

从形态上来看，我们肯定希望找到那些涨势/跌势明显的点。正值很大，表示明显的涨势；正值为 1 以内非常小的值，表示非常明显的跌势。在负值较小的情况下越接近 1，说明涨势越明显；在负值较大的情况下越接近 1（最大就是 1），说明跌势越明显。正值且接近 1 的情况就不太好了，说明涨跌幅度相近，这非常不利于交易。所以，显然是幅度越大且涨跌态势越明确

（不震荡）的情形是最好的下单入场点。

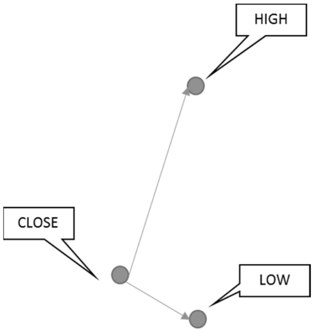


图 18–17　正值较大

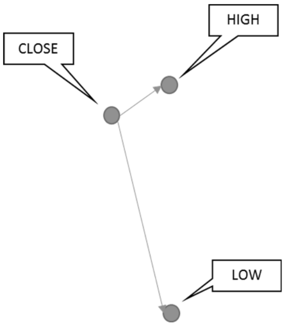


图 18–18　正值较小

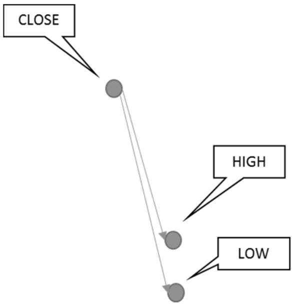


图 18–19　负值较大

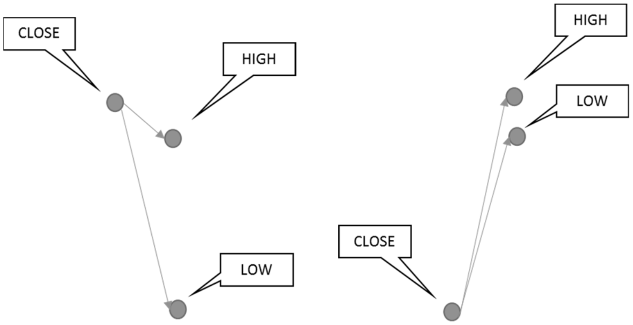


图 18–20　负值较小

通过分析这个文件就会发现，只有正值较大和正值较小两种情况，也就是说，CLOSE 一直都在 HIGH 和 LOW 之间。这种情况下就要看波动向哪边偏斜了，那些偏斜大的是我们比较喜欢的。在 1 小时内升幅最大的是 386 个点，跌幅最大的是 266 个点。我们试着找出那些波动幅度明显属于向高方向偏斜的（这里可以考虑大于 1.5 的），以及波动幅度明显属于向低方向偏斜的（这里可以考虑小于 0.66 的），看看数量分别有多少。

建立一个临时表 SLIDEWINDOW。

```
CREATE TABLE SLIDEWINDOW(
   DT DATETIME,
   RISE_PIP DECIMAL(10,4),
   FALL_PIP DECIMAL(10,4),
   PROPORTION DECIMAL(10,4)
);
```

将分析结果插入这个临时表，编写 Python 文件 slidewindow_insert.py。

```
#!/usr/bin/python
import os,sys,MySQLdb
```

```
try:
  db = MySQLdb.connect(host = 'localhost', user='root', passwd = '111111', db = 'FOREX')

  cursor = db.cursor()
  counter = 0
  cursor.execute('USE FOREX;')
  sql = 'SELECT * FROM EURUSD_1M'
  cursor.execute(sql);
  result = cursor.fetchall()

  for i in range(0, cursor.rowcount):
    startdt = str(result[i][0])
    startpip = str(result[i][4])
    cursor1 = db.cursor()
    cursor1.execute('USE FOREX;')
    sql1 = 'INSERT INTO SLIDEWINDOW SELECT DT, MAX(HIGH)-' + startpip + ' AS RISE_PIP, ' + startpip
+ '-MIN(LOW) AS FALL_PIP, CASE WHEN '+ startpip + '<>MIN(LOW) THEN (MAX(HIGH)-' + startpip + ')/('
+ startpip + '-MIN(LOW)) ELSE (MAX(HIGH)-' + startpip + ')/0.0001 END FROM EURUSD_1M WHERE DT BETWEEN
"' + startdt + '" AND DATE_ADD("' + startdt + '", INTERVAL 60 MINUTE)'
    cursor1.execute(sql1)

    if i%2000==0:
      db.commit()
  db.commit()
except MySQLdb.Error,e:
  print "Error %s" % (str(e.args[0])+':'+str(e.args[1]))
  exit(1)
cursor1.close()
cursor.close()
db.close()
```

使用 SQL 语句，计算涨幅与跌幅波动比大于 1.5 且涨幅大于 10 个点的分钟数。

```
SELECT
   COUNT(*)
FROM
   SLIDEWINDOW
WHERE
   PROPORTION>=1.5 AND RISE_PIP>=0.0010;
```

共找到 1606759 分钟的数据，占 5447760 分钟记录的 29.5%，说起来不算太差。在这种情况下，理论上有超过 $\frac{1}{4}$ 的时间可以下单买涨。

下面计算一下涨幅与跌幅波动比小于 0.66 且跌幅大于 10 个点的分钟数。

```
SELECT
   COUNT(*)
FROM
   SLIDEWINDOW
WHERE
   PROPORTION<=0.66 AND FALL_PIP>=0.0010;
```

共找到 1576827 分钟的数据，占 5447760 分钟记录的 28.9%，与涨势的情况相差无几。

下面我们找出这些点前面的 1M K 线图数据，并尝试设计一个分类模型。

构造训练样本

为了方便训练，我们先构建一个便于生成训练样本的表或者文件。在 Keras 里，训练样本和验证集是以如下这种文本形式出现的。

```
 [[ 0.48797666  0.49515848  0.96743517 ...,  0.34333791  0.01071582
   0.55347333]
 [ 0.18182799  0.52856064  0.7593987  ...,  0.93302057  0.96330937
   0.36177566]
 [ 0.29722546  0.69625588  0.02873254 ...,  0.48557884  0.73657364
   0.32492677]
 ...,
 [ 0.13549382  0.00378319  0.06903272 ...,  0.27581255  0.00568914
   0.32686159]
 [ 0.24844718  0.8345029   0.63331516 ...,  0.15998351  0.56751348
   0.09996194]
 [ 0.33877344  0.37488438  0.42507674 ...,  0.44084842  0.18272158
   0.70054914]]
[[0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 ...,
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]]
```

前半部分最外层的方括号中是一个个输入向量，用方括号括起来，中间的维度用空格分开。后半部分的分类向量信息，最外层用方括号括起来，每一个分类向量也用方括号隔离。在这里，由于使用 Softmax 作为输出层，所以每一个输出向量的每一个维度都是一个分类概率。

使用 SQL 语句建造一个训练表，这个表中既要包含刚刚所有的 EURUSD 表中的信息，也要包括 SLIDEWINDOW 表中的信息。

```
CREATE TABLE EURUSD_TRAINING
SELECT
```

```
    EURUSD_1M.DT,
    OPEN,
    HIGH,
    LOW,
    CLOSE,
    RISE_PIP,
    FALL_PIP,
    PROPORTION
FROM
    EURUSD
INNER JOIN
    SLIDEWINDOW
ON
    EURUSD_1M.DT=SLIDEWINDOW.DT;
```

此外，还要给每条训练记录加上一个分类，就是我们前面提到的 RISE3、RISE2、RISE1、NONE、FALL1、FALL2、FALL3 这几个分类。为了方便起见，最后在向量构成时会把它们对应地改写成 1、2、3、4、5、6、7 这样 7 个分类。在此之前，改造一下 EURUSD_TRAINING 表，给它加上一个 CLASSIFICATION 字段。

```
ALTER TABLE EURUSD_TRAINING ADD COLUMN(CLASSIFICATION INT);
```

在表构造环节的最后，要加上分类条件。

```
UPDATE EURUSD_TRAINING
SET CLASSIFICATION = 1
WHERE RISE_PIP>=0.0040 AND PROPORTION>=1.5;

UPDATE EURUSD_TRAINING
SET CLASSIFICATION = 2
WHERE RISE_PIP>=0.0020 AND RISE_PIP<0.0040 AND PROPORTION>=1.5;

UPDATE EURUSD_TRAINING
SET CLASSIFICATION = 3
WHERE RISE_PIP>=0.0010 AND RISE_PIP<0.0020 AND PROPORTION>=1.5;

UPDATE EURUSD_TRAINING
SET CLASSIFICATION = 5
WHERE FALL_PIP>=0.0010 AND FALL_PIP<0.0020 AND PROPORTION<0.67;

UPDATE EURUSD_TRAINING
SET CLASSIFICATION = 6
WHERE FALL_PIP>=0.0020 AND FALL_PIP<0.0040 AND PROPORTION<0.67;

UPDATE EURUSD_TRAINING
```

```
SET CLASSIFICATION = 7
WHERE FALL_PIP>=0.0040 AND PROPORTION>=1.5 AND PROPORTION<0.67;

UPDATE EURUSD_TRAINING
SET CLASSIFICATION = 4
WHERE CLASSIFICATION IS NULL;

COMMIT;
```

这里要注意最后一个分类，其他几个分类的定义都比较清晰。最后一个分类其实并不仅仅是波动小的分类，还包括一些情况，例如上下波幅可能很大但涨跌幅度相当。这两种分类合起来作为分类“4”，我们将它们统称为“不适合入场”就好了。

编写训练脚本

在滑动窗口捕捉到涨跌信息后，可以对应输出数据分类。由于数据量比较充足，可以考虑用 70% 的数据作为训练数据，留下 30% 的数据作为验证数据。

使用 Keras 和 Theano 搭建一个训练环境，并选择相应的数据项作为训练集和验证集。在这个训练过程中，我们假设从当前的时间点向前观察 60 分钟，结合当前时间点的“蜡烛”位置来判断是否产生了“突破”形态，并使之向着自己想定的方向移动，即今后 1 小时的涨势信息蕴含在之前 60 分钟 K 线图每根“蜡烛”所表示的价位之中。

在做这部分工作之前，我们处理一下 EURUSD_TRAINING 表，为其添加一个索引。

```
ALTER TABLE EURUSD_TRAINING ADD INDEX (DT);
```

在这里请注意，可以直接在数据库里用游标做滑动窗口构造训练样本集。这种情况下，时间复杂度在 $O(n)$ 上。但是，不推荐在遍历中嵌套一个 60 分钟的请求——尽管时间复杂度理论上还是 $O(n)$，不过由于每次都要再请求数据库，所以消耗的时间很长。在这里我使用的方法是将数据输出到文本文件，然后用遍历文本文件的方式进行构造。

编写 Python 文件 make_file.py。

```
import os,sys,MySQLdb
import numpy as np

db = MySQLdb.connect(host = 'localhost', user='root', passwd = '111111', db = 'FOREX')
cursor = db.cursor()
cursor.execute('USE FOREX;')
sql = 'SELECT * FROM EURUSD_TRAINING;'
cursor.execute(sql)
result = cursor.fetchall()

for i in range(cursor.rowcount):
```

```
  print
str(result[i][0])+','+str(result[i][1])+','+str(result[i][2])+','+str(result[i][3])+','+str(r
esult[i][4])+','+str(result[i][5])+','+str(result[i][6])+','+str(result[i][7])+','+str(result
[i][8])

cursor.close()
db.close()
```

调用该文件，并把文件输出到 record.txt。

```
python make_file.py >> record.txt
```

（1）第 1 次尝试

在这个文件中，我们设计了一个有 8 个维度的向量，但这 8 个维度在训练中并非都用得上。编写训练文件 train.py。

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD
import os,sys,time
import numpy as np

print "Start to generate network"
model = Sequential()

model.add(Dense(120, input_dim=240))
model.add(Activation('linear'))
model.add(Dense(60, input_dim=120))
model.add(Activation('linear'))
model.add(Dense(30, input_dim=60))
model.add(Activation('linear'))
model.add(Dense(15, input_dim=30))
model.add(Activation('linear'))
model.add(Dense(7, input_dim=15))
model.add(Activation('softmax'))

sgd = SGD(lr=0.02)

model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

print "start to load data"
records = open('./record_3500000.txt','r')

X_train = []
y_train = []
```

```
line_pointer = -1

for line in records.readlines():
  line_pointer = line_pointer + 1
  X_train.append([])
  y_train.append([])

  values = line.split(',')

  if (line_pointer<=59):
    line_length = line_pointer
  else:
    line_length = 59

  the_time = time.strptime(str(values[0]),"%Y-%m-%d %H:%M:%S")
  X_train[line_pointer].append(float(values[1]))
  X_train[line_pointer].append(float(values[2]))
  X_train[line_pointer].append(float(values[3]))
  X_train[line_pointer].append(float(values[4]))
  #print len(X_train[line_pointer-1])
  for j in range(line_length):
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+4])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+5])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+6])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+7])
  for i in range(240-len(X_train[line_pointer])):
    X_train[line_pointer].append(0)

  for k in range(7):
    y_train[line_pointer].append(0)
  y_train[line_pointer][int(values[8])] = 1

  #print y_train

  if line_pointer%1000==0:
    print line_pointer

print "start training"

model.fit(X_train, y_train, nb_epoch=100, batch_size=2000, validation_split=0.2)

json_string = model.to_json()
open('./my_model_architecture.json', 'w').write(json_string)
model.save_weights('./my_model_weights.h5')
```

在这个文件中，构造的 X_train 就是输入向量，y_train 就是输出向量。X_train 的内容是在每一分钟向过去的方向看 60 分钟，这 60 分钟的 OPEN、HIGH、LOW、CLOSE 一共组成 240 个维度的向量，一共 544 万分钟左右的数据。通过这种构造，会形成一个 544 万 × 240 个维度的二维数组作为训练数据，在文件开始的“空洞”部分会用“0”补齐。由于数量比例非常小，所以这部分数据不会影响训练结果。y_train 是分类标签向量。由于我们分了 7 个类，所以构造出来的是一个 544 万 × 7 的数组。在这种情况下，使用 Crossentropy 作为损失函数更为合适。

由于我的 PC 内存有限，如果把 544 万条数据都放入进行训练，会导致内存溢出，所以最后我只选用了时间最晚的 350 万条数据（将近 10 年的数据），进行如下处理。

```
head -3500000 record.txt >> record_3500000.txt
```

调用该文件。

```
THEANO_FLAGS=device=gpu,floatX=float32 python train.py
```

由于在文件中有进度显示的打印功能，因此会出现如下提示信息。

```
......
3493000
3494000
3495000
3496000
3497000
3498000
3499000
start training
Train on 2800000 samples, validate on 700000 samples
Epoch 1/100
2800000/2800000 [==============================] - 12s - loss: 1.5626 - acc: 0.4333 - val_loss:
1.4892 - val_acc: 0.4720
Epoch 2/100
2800000/2800000 [==============================] - 11s - loss: 1.5589 - acc: 0.4337 - val_loss:
1.4849 - val_acc: 0.4720
Epoch 3/100
2800000/2800000 [==============================] - 11s - loss: 1.5581 - acc: 0.4337 - val_loss:
1.4810 - val_acc: 0.4720
Epoch 4/100
2800000/2800000 [==============================] - 11s - loss: 1.5576 - acc: 0.4337 - val_loss:
1.4811 - val_acc: 0.4720
Epoch 5/100
2800000/2800000 [==============================] - 11s - loss: 1.5575 - acc: 0.4337 - val_loss:
1.4859 - val_acc: 0.4720
Epoch 6/100
```

```
2800000/2800000 [==============================] - 11s - loss: 1.5574 - acc: 0.4337 - val_loss:
1.4843 - val_acc: 0.4720
Epoch 7/100
2800000/2800000 [==============================] - 11s - loss: 1.5573 - acc: 0.4337 - val_loss:
1.4855 - val_acc: 0.4720
Epoch 8/100
2800000/2800000 [==============================] - 11s - loss: 1.5573 - acc: 0.4337 - val_loss:
1.4847 - val_acc: 0.4720
Epoch 9/100
2800000/2800000 [==============================] - 11s - loss: 1.5573 - acc: 0.4337 - val_loss:
1.4871 - val_acc: 0.4720
Epoch 10/100
2800000/2800000 [==============================] - 11s - loss: 1.5572 - acc: 0.4337 - val_loss:
1.4852 - val_acc: 0.4720
……
Epoch 91/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4898 - val_acc: 0.4720
Epoch 92/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4910 - val_acc: 0.4720
Epoch 93/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4910 - val_acc: 0.4720
Epoch 94/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4914 - val_acc: 0.4720
Epoch 95/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4897 - val_acc: 0.4720
Epoch 96/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4897 - val_acc: 0.4720
Epoch 97/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4895 - val_acc: 0.4720
Epoch 98/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4889 - val_acc: 0.4720
Epoch 99/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4886 - val_acc: 0.4720
Epoch 100/100
2800000/2800000 [==============================] - 11s - loss: 1.5568 - acc: 0.4337 - val_loss:
1.4901 - val_acc: 0.4720
```

从训练的情况来看，在验证集上损失函数的值降到 1.4900 附近，验证集上的准确率下降到 47.20% 左右就不再下降了。

这种 *Loss* 不再下降的情况，一般来说有相对固定的套路去解决，一是调整网络参数，二是加入更多维度。调整网络参数主要是指网络结构、激励函数和损失函数的调整。加入维度是指向量信息的丰富。目前加入的只是 60 分钟的裸 K 线数据，没有其他信息。基于货币对的周期性波动规律假说，我们在下一次的尝试中可以加入月、星期、小时 3 个维度。

（2）第 2 次尝试

重新设计网络。

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD
import os,sys,time
import numpy as np

print "Start to generate network"
model = Sequential()
model.add(Dense(120, input_dim=243))
model.add(Activation('sigmoid'))
model.add(Dense(60, input_dim=120))
model.add(Activation('sigmoid'))
model.add(Dense(30, input_dim=60))
model.add(Activation('sigmoid'))
model.add(Dense(15, input_dim=30))
model.add(Activation('sigmoid'))
model.add(Dense(7, input_dim=15))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

print "start to load data"

records = open('./record_3500000.txt','r')

X_train = []
y_train = []

line_pointer = -1

for line in records.readlines():
  line_pointer = line_pointer + 1
  X_train.append([])
  y_train.append([])
```

```
  values = line.split(',')

  if (line_pointer<=59):
    line_length = line_pointer
  else:
    line_length = 59

  the_time = time.strptime(str(values[0]),"%Y-%m-%d %H:%M:%S")
  X_train[line_pointer].append(float(time.strftime("%H",the_time)))
  X_train[line_pointer].append(float(time.strftime("%m",the_time)))
  X_train[line_pointer].append(float(time.strftime("%w",the_time)))
  X_train[line_pointer].append(float(values[1]))
  X_train[line_pointer].append(float(values[2]))
  X_train[line_pointer].append(float(values[3]))
  X_train[line_pointer].append(float(values[4]))

  for j in range(line_length):
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+4])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+5])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+6])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+7])
  for i in range(243-len(X_train[line_pointer])):
    X_train[line_pointer].append(0)

  for k in range(7):
    y_train[line_pointer].append(0)
  y_train[line_pointer][int(values[8])] = 1

  if line_pointer%1000==0:
    print line_pointer

print "start training"

model.fit(X_train, y_train, nb_epoch=100, batch_size=2000, validation_split=0.15)

json_string = model.to_json()
open('./my_model_architecture.json', 'w').write(json_string)
model.save_weights('./my_model_weights.h5')
```

这次调整把激励函数改成了 Sigmoid 函数，目的是引入更多的非线性特征。

```
the_time = time.strptime(str(values[0]),"%Y-%m-%d %H:%M:%S")
X_train[line_pointer].append(float(time.strftime("%H",the_time)))
X_train[line_pointer].append(float(time.strftime("%m",the_time)))
X_train[line_pointer].append(float(time.strftime("%w",the_time)))
```

这个部分就是加入了月、星期、小时 3 个维度后再进行训练。

```
……
Train on 2975000 samples, validate on 525000 samples
Epoch 1/100
2975000/2975000 [==============================] - 15s - loss: 1.5305 - acc: 0.4356 - val_loss:
1.4402 - val_acc: 0.4623
Epoch 2/100
2975000/2975000 [==============================] - 13s - loss: 1.5071 - acc: 0.4377 - val_loss:
1.4264 - val_acc: 0.4623
Epoch 3/100
2975000/2975000 [==============================] - 13s - loss: 1.5047 - acc: 0.4377 - val_loss:
1.4297 - val_acc: 0.4623
Epoch 4/100
2975000/2975000 [==============================] - 13s - loss: 1.5030 - acc: 0.4377 - val_loss:
1.4218 - val_acc: 0.4623
Epoch 5/100
2975000/2975000 [==============================] - 13s - loss: 1.5014 - acc: 0.4377 - val_loss:
1.4288 - val_acc: 0.4623
Epoch 6/100
2975000/2975000 [==============================] - 13s - loss: 1.5001 - acc: 0.4377 - val_loss:
1.4178 - val_acc: 0.4623
Epoch 7/100
2975000/2975000 [==============================] - 13s - loss: 1.4991 - acc: 0.4377 - val_loss:
1.4165 - val_acc: 0.4623
Epoch 8/100
2975000/2975000 [==============================] - 13s - loss: 1.4983 - acc: 0.4377 - val_loss:
1.4335 - val_acc: 0.4623
Epoch 9/100
2975000/2975000 [==============================] - 13s - loss: 1.4976 - acc: 0.4377 - val_loss:
1.4125 - val_acc: 0.4623
Epoch 10/100
2975000/2975000 [==============================] - 13s - loss: 1.4969 - acc: 0.4377 - val_loss:
1.4089 - val_acc: 0.4623
……
Epoch 91/100
2975000/2975000 [==============================] - 13s - loss: 1.4645 - acc: 0.4380 - val_loss:
1.4858 - val_acc: 0.4405
Epoch 92/100
2975000/2975000 [==============================] - 13s - loss: 1.4642 - acc: 0.4379 - val_loss:
1.4997 - val_acc: 0.4252
Epoch 93/100
2975000/2975000 [==============================] - 13s - loss: 1.4640 - acc: 0.4380 - val_loss:
1.4827 - val_acc: 0.4447
Epoch 94/100
2975000/2975000 [==============================] - 13s - loss: 1.4638 - acc: 0.4379 - val_loss:
```

```
1.4780 - val_acc: 0.4330
Epoch 95/100
2975000/2975000 [==============================] - 13s - loss: 1.4635 - acc: 0.4380 - val_loss:
1.4947 - val_acc: 0.4308
Epoch 96/100
2975000/2975000 [==============================] - 13s - loss: 1.4633 - acc: 0.4380 - val_loss:
1.5014 - val_acc: 0.4367
Epoch 97/100
2975000/2975000 [==============================] - 13s - loss: 1.4631 - acc: 0.4380 - val_loss:
1.4762 - val_acc: 0.4432
Epoch 98/100
2975000/2975000 [==============================] - 13s - loss: 1.4629 - acc: 0.4379 - val_loss:
1.4838 - val_acc: 0.4398
Epoch 99/100
2975000/2975000 [==============================] - 13s - loss: 1.4629 - acc: 0.4379 - val_loss:
1.4888 - val_acc: 0.4400
Epoch 100/100
```

损失函数的值比第 1 次设计的要小，但准确率没有提升。这说明月、星期、小时 3 个参考值的加入对模型精度的提升没有帮助。

（3）第 3 次尝试

通过前两次尝试我们发现，这个分类过程对熵减没有太多帮助。为什么这么说呢？如果用训练出来的模型进行回测，会发现绝大部分的样本都集中在波幅最小的那一部分，这一点在下面更细的定量分析中就能看出来。所以，如果希望熵减更为明显，就要引入更多的维度，而且尽量不要将分类设置得太“倾斜”，也就是尽量不要让其中个别分类所占的比例太大。

我们思考一下：既然预测 1 小时的情况可能会比较复杂，那么是不是可以考虑将这个时间缩短呢？因为时间越长，混沌所产生的影响就越明显；熵越大，预测难度也就越大。这一次我们试试用 15 分钟内的波幅来分析。

建立表 SLIDEWINDOW_15M。

```
CREATE TABLE SLIDEWINDOW_15M(
DT DATETIME,
RISE_PIP DECIMAL(10,4),
FALL_PIP DECIMAL(10,4),
PROPORTION DECIMAL(10,4)
);
```

将分析结果插入这个临时表，编写 Python 文件 slidewindow_15m_insert.py。

```
#!/usr/bin/python
import os,sys,MySQLdb
try:
```

```
  db = MySQLdb.connect(host = 'localhost', user='root', passwd = '111111', db = 'FOREX')

  cursor = db.cursor()
  counter = 0
  cursor.execute('USE FOREX;')
  sql = 'SELECT * FROM EURUSD_1M'
  cursor.execute(sql);
  result = cursor.fetchall()

  for i in range(0, cursor.rowcount):
    startdt = str(result[i][0])
    startpip = str(result[i][4])
    cursor1 = db.cursor()
    cursor1.execute('USE FOREX;')
    sql1 = 'INSERT INTO SLIDEWINDOW_15M SELECT DT, MAX(HIGH)-' + startpip + ' AS RISE_PIP, ' + startpip + '-MIN(LOW) AS FALL_PIP, CASE WHEN '+ startpip + '<>MIN(LOW) THEN (MAX(HIGH)-' + startpip + ')/(' + startpip + '-MIN(LOW)) ELSE (MAX(HIGH)-' + startpip + ')/0.0001 END FROM EURUSD_1M WHERE DT BETWEEN "' + startdt + '" AND DATE_ADD("' + startdt + '", INTERVAL 15 MINUTE)'
    cursor1.execute(sql1)

    if i%1000==0:
      db.commit()
  db.commit()
except MySQLdb.Error,e:
  print "Error %s" % (str(e.args[0])+':'+str(e.args[1]))
  exit(1)
cursor1.close()
cursor.close()
db.close()
```

在这里，我们填充 SLIDESHOW_15M 表的逻辑和填充 SLIDESHOW 表的逻辑相同，只是向后看的周期调整到了 15 分钟，其他逻辑则没有没变化。

建立表 EURUSD_TRAINING_15M。

```
CREATE TABLE EURUSD_TRAINING_15M
SELECT
   EURUSD_1M.DT,
   OPEN,
   HIGH,
   LOW,
   CLOSE,
   RISE_PIP,
   FALL_PIP,
   PROPORTION
   FROM
```

```
    EURUSD_1M
INNER JOIN
    SLIDEWINDOW
ON
EURUSD_1M.DT=SLIDEWINDOW.DT;
```

F 和 R 两个字段分别作为跌幅 FALL 和涨幅 RISE 的分类标记，后面很快会用到。

在进行分类标注之前，我们看一下这 15 分钟内的涨跌数据都在什么范围内集中。

```
mysql> SELECT COUNT(*), F, R FROM (SELECT FALL_PIP DIV 0.0010 AS F, RISE_PIP DIV 0.0010 AS R FROM
EURUSD_TRAINING_15M) R GROUP BY F, R;
+----------+------+------+
| COUNT(*) | F    | R    |
+----------+------+------+
|  3920354 |    0 |    0 |
|   571240 |    0 |    1 |
|   112985 |    0 |    2 |
|    30347 |    0 |    3 |
|    10061 |    0 |    4 |
|     3834 |    0 |    5 |
|     1605 |    0 |    6 |
|      858 |    0 |    7 |
|      411 |    0 |    8 |
|      260 |    0 |    9 |
|      169 |    0 |   10 |
|      ……
|        6 |   16 |    0 |
|        3 |   16 |    1 |
|       14 |   17 |    0 |
|        1 |   17 |    1 |
|        2 |   18 |    0 |
|        2 |   19 |    1 |
|        1 |   20 |    0 |
|        1 |   20 |    1 |
|        1 |   21 |    0 |
|        1 |   21 |    1 |
|        1 |   22 |    0 |
|        1 |   22 |    7 |
|        1 |   23 |    2 |
+----------+------+------+
179 rows in set (4.14 sec)
```

这是用 10 个点做分割的最小区间来做的，涨跌幅度看后面的 F 和 R 就知道了。例如，“| 3920354 | 0 | 0 |” 这一条，表示涨跌都在 10 个点以内的振幅小且方向不确定的数据超过

392 万条，占全部样本的 72%。不得不说，这个比例实在是太大了，相当于其他 178 个分类总共才占 28%——这是一个非常“倾斜”的分类方式。

我们可以尝试再做一下细分，用 5 个点作为一个分档。

```
mysql> SELECT COUNT(*), F, R FROM (SELECT FALL_PIP DIV 0.0005 AS F, RISE_PIP DIV 0.0005 AS R FROM
EURUSD_TRAINING_15M) R GROUP BY F, R;
+----------+------+------+
| COUNT(*) | F    | R    |
+----------+------+------+
|  1916865 |    0 |    0 |
|   895584 |    0 |    1 |
|   334411 |    0 |    2 |
|   134729 |    0 |    3 |
|    60612 |    0 |    4 |
|    29028 |    0 |    5 |
……
|   894166 |    1 |    0 |
|   213739 |    1 |    1 |
|    71067 |    1 |    2 |
|    31033 |    1 |    3 |
|    15198 |    1 |    4 |
……
|        4 |   34 |    0 |
|        1 |   34 |    1 |
|        1 |   34 |    3 |
|        9 |   35 |    0 |
|        1 |   37 |    0 |
|        1 |   37 |    1 |
|        1 |   38 |    2 |
|        1 |   38 |    3 |
|        1 |   40 |    0 |
|        1 |   40 |    2 |
|        1 |   42 |    1 |
|        1 |   42 |    3 |
|        1 |   44 |    0 |
|        1 |   44 |   14 |
|        1 |   46 |    4 |
+----------+------+------+
513 rows in set (4.31 sec)
```

在这个划分中，虽然涨跌都在 5 个点的分类“|　1916865 |　　0 |　　0 |”仍然有 191 万条记录，但是占比已经下降到 35% 左右。

下面，我们用其他处理技巧作为辅助，继续这个熵减过程。

```
ALTER TABLE EURUSD_TRAINING_15M ADD COLUMN(F INT);

ALTER TABLE EURUSD_TRAINING_15M ADD COLUMN(R INT);

UPDATE EURUSD_TRAINING_15M SET R=RISE_PIP DIV 0.0005, F=FALL_PIP DIV 0.0005;

COMMIT;
```

要想给 EURUSD_TRAINING_15M 这个表上加上标签，可以先画一个表，看看它们的分布是什么样子（如图 18-21 所示）。

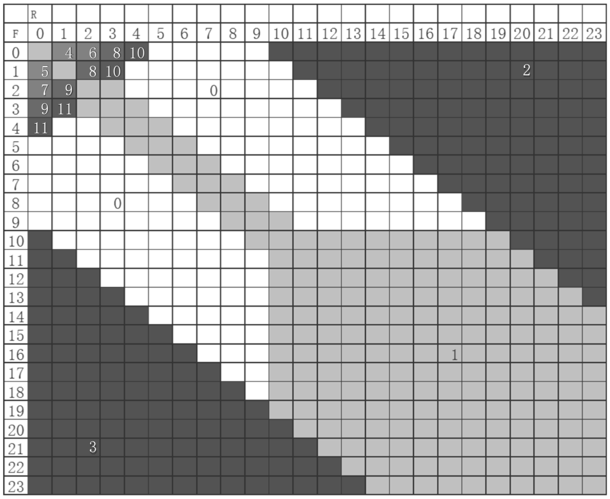


图 18-21　分类分布

我是这样做的：把 F 和 R 两种数值在二维表格中画出来，在没有做任何分类之前，整个表格出现的都是“0”这个类别，即熵最大的情况；“1”是对角线的部分，就是 F 和 R 振幅相当的情况；右下侧的部分是振幅很大（虽然两侧振幅不相等，但是幅度很大）的分类，明显是震荡盘面，不适合下单；“2”和“3”是单边振幅明显较大的分类，“2”是上涨，“3”是下跌；“4”和“5”分别是涨幅 6 ~ 10 个点与跌幅 0 ~ 5 个点的分类、跌幅 0 ~ 5 个点与涨幅 6 ~ 10 个点的分类，仍属于单方面比较确定的方向，只不过是幅度很小的情况。

对其他分类，分别按照其 R 和 F 的标识即可进行标注。从理论上说，除了“0”和“1”两个分类以外，其他的分类都可以尝试入场。

用 SQL 做一下更新。

```
#WHITE
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 0;
```

```
#YELLOW
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 1
WHERE R>=10 AND F>=10;

UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 1
WHERE ABS(R-F)<=1;

#RED-R
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 2
WHERE R>=F+10;

#RED-F
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 3
WHERE F>=R+10;

#CYAN-R
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 4
WHERE R=1 AND F=0;

#CYAN-F
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 5
WHERE R=0 AND F=1;

#GREEN-R
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 6
WHERE R=2 AND F=0;

#GREEN-F
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 7
WHERE R=0 AND F=2;

#BLUE-R
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 8
WHERE R=3 AND F=0;

UPDATE EURUSD_TRAINING_15M
```

```
SET CLASSIFICATION = 8
WHERE R=2 AND F=1;

#BLUE-F
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 9
WHERE R=0 AND F=3;

UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 9
WHERE R=1 AND F=2;

#PURPLE-R
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 10
WHERE R=4 AND F=0;

UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 10
WHERE R=3 AND F=1;

#PURPLE-F
UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 11
WHERE R=0 AND F=4;

UPDATE EURUSD_TRAINING_15M
SET CLASSIFICATION = 11
WHERE R=1 AND F=3;
```

将这些训练样本导出为文本，编辑文件 make_file_15m.py。

```
import os,sys,MySQLdb
import numpy as np

db = MySQLdb.connect(host = 'localhost', user='root', passwd = '111111', db = 'FOREX')
cursor = db.cursor()
cursor.execute('USE FOREX;')
sql = 'SELECT * FROM EURUSD_TRAINING_15M;'
cursor.execute(sql)
result = cursor.fetchall()

for i in range(cursor.rowcount):
  print
str(result[i][0])+','+str(result[i][1])+','+str(result[i][2])+','+str(result[i][3])+','+str(r
esult[i][4])+','+str(result[i][5])+','+str(result[i][6])+','+str(result[i][7])+','+str(result
```

```
[i][8])

cursor.close()
db.close()
```

在 Shell 下调用。

```
python make_file_15m.py >> record_15M.txt
```

截断文件，只保留最后的 350 万条记录。

```
tail -3500000 record_15M.txt >> record_15M_3500000.txt
```

编辑训练文件 train_1.py。

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD
import os,sys,time
import numpy as np

print "Start to generate network"
model = Sequential()
model.add(Dense(120, input_dim=63))
model.add(Activation('sigmoid'))
model.add(Dense(80, input_dim=100))
model.add(Activation('sigmoid'))
model.add(Dense(60, input_dim=70))
model.add(Activation('sigmoid'))
model.add(Dense(50, input_dim=120))
model.add(Activation('sigmoid'))
model.add(Dense(40, input_dim=50))
model.add(Activation('sigmoid'))
model.add(Dense(30, input_dim=40))
model.add(Activation('sigmoid'))
model.add(Dense(12, input_dim=30))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

print "start to load data"

records = open('./record_15M_3500000.txt','r')

X_train = []
y_train = []
```

```
line_pointer = -1

for line in records.readlines():
  line_pointer = line_pointer + 1
  X_train.append([])
  y_train.append([])

  values = line.split(',')

  if (line_pointer<=14):
    line_length = line_pointer
  else:
    line_length = 14

  the_time = time.strptime(str(values[0]),"%Y-%m-%d %H:%M:%S")
  X_train[line_pointer].append(float(time.strftime("%H",the_time)))
  X_train[line_pointer].append(float(time.strftime("%m",the_time)))
  X_train[line_pointer].append(float(time.strftime("%w",the_time)))
  X_train[line_pointer].append(float(values[1]))
  X_train[line_pointer].append(float(values[2]))
  X_train[line_pointer].append(float(values[3]))
  X_train[line_pointer].append(float(values[4]))

  for j in range(line_length):
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+4])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+5])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+6])
    X_train[line_pointer].append(X_train[line_pointer-1][j*4+7])
  for i in range(63-len(X_train[line_pointer])):
    X_train[line_pointer].append(0)

  for k in range(12):
    y_train[line_pointer].append(0)
  y_train[line_pointer][int(values[8])] = 1

  if line_pointer%10000==0:
    print line_pointer

print "start training"

#print X_train[0]
#print X_train[100]

model.fit(X_train, y_train, nb_epoch=20, batch_size=2000, validation_split=0.15)
```

```
json_string = model.to_json()
open('./my_model_architecture_1.json', 'w').write(json_string)
model.save_weights('./my_model_weights_1.h5')

pre = model.predict(X_train)

predicted = np.zeros((12,12))

for i in range(len(pre)):
  max_train = 0
  max_pre = 0
  for m in range(12):
    if (y_train[i][m] ==1):
      max_train = m
  for m in range(12):
    if (pre[i][max_pre] < pre[i][m]):
      max_pre = m
  predicted[max_train][max_pre] = predicted[max_train][max_pre] + 1

for i in range(12):
  for j in range(12):
    print predicted[i][j],
  print ""
```

这段代码只执行了 20 轮，执行后会输出下面的内容。

```
Train on 2975000 samples, validate on 525000 samples
Epoch 1/20
2975000/2975000 [==============================] - 14s - loss: 1.8795 - acc: 0.3742 - val_loss:
1.6941 - val_acc: 0.4185
Epoch 2/20
2975000/2975000 [==============================] - 14s - loss: 1.8547 - acc: 0.3751 - val_loss:
1.6928 - val_acc: 0.4185
Epoch 3/20
2975000/2975000 [==============================] - 14s - loss: 1.8531 - acc: 0.3751 - val_loss:
1.6813 - val_acc: 0.4185
Epoch 4/20
2975000/2975000 [==============================] - 14s - loss: 1.8504 - acc: 0.3751 - val_loss:
1.6750 - val_acc: 0.4185
Epoch 5/20
2975000/2975000 [==============================] - 14s - loss: 1.8476 - acc: 0.3750 - val_loss:
1.6702 - val_acc: 0.4185
Epoch 6/20
2975000/2975000 [==============================] - 14s - loss: 1.8458 - acc: 0.3750 - val_loss:
1.6696 - val_acc: 0.4180
Epoch 7/20
```

```
2975000/2975000 [==============================] - 14s - loss: 1.8448 - acc: 0.3751 - val_loss:
1.6637 - val_acc: 0.4187
Epoch 8/20
2975000/2975000 [==============================] - 14s - loss: 1.8439 - acc: 0.3752 - val_loss:
1.6783 - val_acc: 0.4174
Epoch 9/20
2975000/2975000 [==============================] - 14s - loss: 1.8428 - acc: 0.3752 - val_loss:
1.6555 - val_acc: 0.4186
Epoch 10/20
2975000/2975000 [==============================] - 14s - loss: 1.8415 - acc: 0.3752 - val_loss:
1.6528 - val_acc: 0.4186
Epoch 11/20
2975000/2975000 [==============================] - 14s - loss: 1.8405 - acc: 0.3752 - val_loss:
1.6534 - val_acc: 0.4185
Epoch 12/20
2975000/2975000 [==============================] - 14s - loss: 1.8398 - acc: 0.3753 - val_loss:
1.6525 - val_acc: 0.4185
Epoch 13/20
2975000/2975000 [==============================] - 14s - loss: 1.8391 - acc: 0.3753 - val_loss:
1.6504 - val_acc: 0.4186
Epoch 14/20
2975000/2975000 [==============================] - 14s - loss: 1.8384 - acc: 0.3754 - val_loss:
1.6573 - val_acc: 0.4186
Epoch 15/20
2975000/2975000 [==============================] - 14s - loss: 1.8376 - acc: 0.3754 - val_loss:
1.6475 - val_acc: 0.4185
Epoch 16/20
2975000/2975000 [==============================] - 14s - loss: 1.8366 - acc: 0.3754 - val_loss:
1.6498 - val_acc: 0.4186
Epoch 17/20
2975000/2975000 [==============================] - 14s - loss: 1.8348 - acc: 0.3754 - val_loss:
1.6688 - val_acc: 0.4168
Epoch 18/20
2975000/2975000 [==============================] - 14s - loss: 1.8321 - acc: 0.3756 - val_loss:
1.6759 - val_acc: 0.4167
Epoch 19/20
2975000/2975000 [==============================] - 13s - loss: 1.8280 - acc: 0.3760 - val_loss:
1.6991 - val_acc: 0.4127
Epoch 20/20
2975000/2975000 [==============================] - 13s - loss: 1.8239 - acc: 0.3762 - val_loss:
1.7493 - val_acc: 0.4077
27434.0 144944.0 0.0 0.0 0.0 197.0 0.0 0.0 0.0 0.0 0.0 0.0
28574.0 1305778.0 0.0 0.0 9.0 1268.0 0.0 0.0 0.0 0.0 0.0 0.0
1115.0 5092.0 0.0 0.0 0.0 5.0 0.0 0.0 0.0 0.0 0.0 0.0
1067.0 5173.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
17999.0 533833.0 0.0 0.0 5.0 1337.0 0.0 0.0 0.0 0.0 0.0 0.0
```

```
17493.0 528829.0 0.0 0.0 36.0 1471.0 0.0 0.0 0.0 0.0 0.0 0.0
13725.0 205288.0 0.0 0.0 0.0 728.0 0.0 0.0 0.0 0.0 0.0 0.0
13521.0 202727.0 0.0 0.0 10.0 713.0 0.0 0.0 0.0 0.0 0.0 0.0
15244.0 135881.0 0.0 0.0 0.0 375.0 0.0 0.0 0.0 0.0 0.0 0.0
14773.0 134573.0 0.0 0.0 0.0 416.0 0.0 0.0 0.0 0.0 0.0 0.0
8522.0 61368.0 0.0 0.0 0.0 125.0 0.0 0.0 0.0 0.0 0.0 0.0
8434.0 61777.0 0.0 0.0 0.0 141.0 0.0 0.0 0.0 0.0 0.0 0.0
```

为了展示得清晰一些，我们把这个表格放到 Excel 里面去，如图 18–22 所示。

	0	1	2	3	4	5	6	7	8	9	10	11	
0	27434	144944	0	0	0	197	0	0	0	0	0	0	
1	28574	1305778	0	0	9	1268	0	0	0	0	0	0	
2	1115	5092	0	0	0	5	0	0	0	0	0	0	
3	1067	5173	0	0	0	0	0	0	0	0	0	0	
4	17999	533833	0	0	5	1337	0	0	0	0	0	0	
5	17493	528829	0	0	36	1471	0	0	0	0	0	0	
6	13725	205288	0	0	0	728	0	0	0	0	0	0	
7	13521	202727	0	0	10	713	0	0	0	0	0	0	
8	15244	135881	0	0	0	375	0	0	0	0	0	0	
9	14773	134573	0	0	0	416	0	0	0	0	0	0	
10	8522	61368	0	0	0	125	0	0	0	0	0	0	
11	8434	61777	0	0	0	141	0	0	0	0	0	0	1334688

图 18–22　20 轮训练

纵轴表示训练集中的类别标示，横轴表示实际预测时得到的类别标识。这样，每个单元格的含义就很明确了。以纵轴为“2”、横轴为“1”的 5092 这个单元格为例，表示有 5092 个原本被标记为“2”类在模型做判断的过程中被判断为“1”类的误判情况。而横轴为“1”、纵轴为“1”的“1305778”的含义是，有 1305778 个样本应为“1”类，并被判断为“1”类，属于正确识别的情况。理论上讲，应该是对角线上深色单元格中的数值越大越好。在这个模型中，经过 20 轮迭代后，判断正确的样本数量为 1334688 个，只占约 38%。在这个阶段，可以通过增大轮数让算法继续收敛。

为了让收敛速度加快，可以使用 ReLU 作为激励函数。修改这个部分。

```
model = Sequential()
model.add(Dense(120, input_dim=63))
model.add(Activation('relu'))
model.add(Dense(80, input_dim=100))
model.add(Activation('relu'))
model.add(Dense(60, input_dim=70))
model.add(Activation('relu'))
model.add(Dense(50, input_dim=120))
model.add(Activation('relu'))
model.add(Dense(40, input_dim=50))
model.add(Activation('relu'))
model.add(Dense(30, input_dim=40))
model.add(Activation('relu'))
model.add(Dense(12, input_dim=30))
model.add(Activation('softmax'))
```

将训练的轮数变为 150 轮，并将验证集比例调整到 35%。

```
model.fit(X_train, y_train, nb_epoch=150, batch_size=2000, validation_split=0.35)
```

在训练结束后，处理一下输出数据，如图 18–23 所示。

	0	1	2	3	4	5	6	7	8	9	10	11	
0	60622	98994	0	0	2479	2334	544	140	4230	2560	671	1	
1	93446	1107109	0	0	33799	46289	13733	2651	24046	11250	3256	50	
2	2446	3249	0	0	86	85	28	0	163	106	49	0	
3	2543	3239	0	0	58	94	26	23	143	106	8	0	
4	46518	442354	0	0	16743	19931	5332	1073	13064	6444	1681	34	
5	47409	434873	0	0	16900	21003	5431	962	13299	6398	1525	29	
6	27788	165287	0	0	7103	7296	1774	254	6165	3288	732	54	
7	28000	161722	0	0	6988	7660	2192	360	6152	3288	595	14	
8	30385	104744	0	0	4086	3756	858	191	4422	2480	571	7	
9	29995	103354	0	0	3972	3780	1020	199	4196	2778	460	8	
10	18219	44761	0	0	1551	1411	358	80	2117	1206	310	2	
11	18383	45299	0	0	1317	1420	334	54	2021	1261	253	10	1215131

图 18–23　150 轮训练

这次训练的数据变化很明显。在这个模型中，从模型标记的角度来看，应该是“0”和“1”以外的类别识别正确率越高越好。尽管结果还是很不理想，但比只进行 20 轮训练时已经有了进步，熵减的方向没错——让分类结果向对角线集中。

4. 未尽步骤

在这个实践中仍有一些步骤没有进行，这也是沿着数据科学的试探步骤逐步深入的过程。

（1）引入更多的维度对这个模型进行改进。如果是基于压力均线的理论，还需要引入更久以前的数据，而不只是前 15 分钟的数据。此外，可以尝试采取其他分类方式或者引入特征值。例如，可以尝试使用其他分类的分界点，分界点的位置和分类的类别数量一定会影响分类效果，在刚刚的第 3 次尝试中，我们已经看到了实例。对于特征值的引入，例如我们在第 2 次尝试中加入的“小时”、“周”、“月” 3 个特征，虽然没有看到明显的效果，但仍然可以继续尝试，甚至尝试引入其他相关指标。

（2）尝试调整不同周期的观察尺度。理论上说，建议缩短观察周期，如果能拿到秒级的数据，就可以通过尝试观察 1 分钟之内的变化及未来 1 分钟之内的波幅来做分类。交易高频化应该是自动交易的趋势，毕竟高频的交易容易产生快速验证，而且也能发挥短时交易中机器在下单与平仓方面的效率优势。

（3）在模型训练基本成熟之后，就可以考虑在 MT4 平台上用 MQL 语言编写脚本，进行历史数据回测。在历史数据回测基本满足要求之后，就可以考虑在虚拟仓上进行交易。如果仍然没有发现问题，就可以考虑进行实仓交易了。对每一单交易，都要根据历史 K 线统计数据，设置相对较低的止盈和止损位置。

（4）为了降低真实交易的风险，还要考虑在多平台、多个货币对上分别训练模型。将大额资金投入分摊到多个平台、多个货币对的交易中去，以降低平台级风险及由单个货币对产生的波动风险。

在自动交易系统的研究路上，还有很多需要讨论的内容。例如，新闻中的文字和走势的关系也可以通过相应的算法引入模型，对模型分类的准确性进行提升。

18.1.11 小结

K 线图的分析方法其实很多，而且 K 线图不仅可以像刚刚这样通过形态方式来分析，也可以结合任何其他维度来分析。数据科学的魅力就在这里——既然是一种可以被人们掌握的学科，就存在方法论，只要按照这样的方法论用科学的方法逐步验证，几乎任何问题都可以产生数字性描述。

到现在为止，所有单凭 K 线图无法准确预测市场价格的情况在我看来都是由于参与计算的维度不足导致的，而这些维度的参与本身所要花费的成本又相当高。这应该就是大量热衷中长期交易的人及投资家喜欢根据局势和大的市场环境来配置多头和空头的重要原因。

作为一个带有学术性的问题，对 K 线图的研究可以不计成本地、无限地扩展下去。不过，如果是专业的投资人士或者寻求中长期稳定收益的人士，还是要多去关注各国比较重要的经济新闻，尤其是各国央行对本国货币的态度，以及各国之间的贸易动态。通常在这个层面产生的对货币、黄金、股票价格的影响，其幅度、周期，以及影响的确定性和稳定性，都比单纯对 K 线图的分析要准确得多。

18.2 数学能救命

下面这个故事来自第二次世界大战的经典案例。

从 1939 年 9 月 1 日希特勒“闪击”波兰开始，世界大国都被一步一步拖入第二次世界大战的泥潭之中。由于法西斯德国强大的军事实力，战争形势急转直下。1940 年 5 月 25 日，英法联军进行了著名的“敦刻尔克大撤退”（如图 18–24 所示）。虽然有 2.8 万名盟军士兵阵亡，4 万多名盟军士兵被俘，但还是有 33 万多名英法联军士兵从欧洲大陆渡过英吉利海峡，北撤至大不列颠岛，为将来固守孤岛甚至进行战略反扑保存了宝贵的有生力量。

图 18–24　敦刻尔克大撤退

18.2.1　阴云下的大西洋

在当时，对战略资源匮乏的大不列颠岛来说，几乎面临着灭顶之灾。岛上的绝大多数战略资源，例如汽油、武器、飞机、粮食、被服等，都严重依赖一条战争补给线艰难地维持，这就是北大西洋航线——唯一的生命线。这条生命线的一端是美国和加拿大的东海岸，另一端是大不列颠岛的西海岸，每天有大量商船来往，为饱受创伤的英国输送“血液”。但是，这条“血管”被德军盯上了。

德军为了切断英军的“大动脉”也下了不少功夫。德军使用最为得心应手的就是被誉为“狼群”的 U 型潜艇群（如图 18–25 所示）。客观地说，U 型潜艇算不上当时世界上性能最为先进的战术潜艇，但由于它出没较为隐蔽，并且在发现“猎物”以后能够快速集合作战，尤其是它狙击的目标是没有反潜能力的商船，所以让盟军的商船船队吃尽了苦头（如图 18–26 所示）。

图 18–25　U 型潜艇

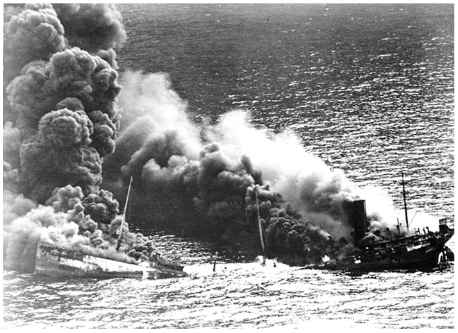


图 18–26　被 U 型潜艇的鱼雷击沉的盟军商船

1939 年 9 月 3 日，也就是英国对德国宣战的数小时后，德国的 U30 潜艇就击沉了一艘名为“雅典娜号”的 1.35 万吨的客轮，客轮上有 1103 名平民，其中包括为了逃避战乱从英国逃往美国的美国人。

1940 年 6 月，U 型潜艇击沉英国商船及护航舰艇 58 艘，共计 28.4 万吨。同年 7 月到 10 月，U 型潜艇共击沉 205 艘英国舰船，共计 108.9 万吨。

1941 年，德国共击沉盟国船只 432 艘，共计 217 万吨。

1942 年，仅当年最后 3 个月，德国共击沉 168 万吨的盟国商船。1942 年全年，盟国船只被击沉 1160 艘，总吨位达 626.6 万吨，超过了当时英美建造新舰的总吨位数。

18.2.2　护航船队的救星

威廉·肖克利[136]（如图 18-27 所示）于 1956 年发明了晶体管，并获得诺贝尔奖。肖克利是

136　威廉·肖克利（William Shockley，1910 年 2 月 ~ 1989 年），在贝尔实验室期间与人共同发明晶体管，被媒体和科学界称为“20 世纪最重要的发明”，和另两位同事一起荣获 1956 年度诺贝尔物理学奖。

一位著名的物理学家。20 世纪 40 年代，年轻的肖克利加入了一个由英国及美国科学家组成的团队，在战争史中开创了使用数据引导战争并颠覆战争进程的先河。

这段渐渐隐去的插曲记录在 Stephen Budiansky 的 *Blackett's War* 一书中。这本书讲述了一个由厌战的数学家、物理学家组成的小团体，通过专心的审查，对数据进行收集和整理，并对汇总的信息进行分析，最终说服盟军海军和空军领导人放下了他们教条的、根深蒂固的偏见，并改变了他们的战斗方式（标志着运筹学的诞生）的故事。

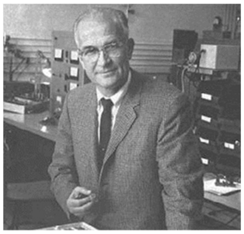


图 18-27　威廉 · 肖克利

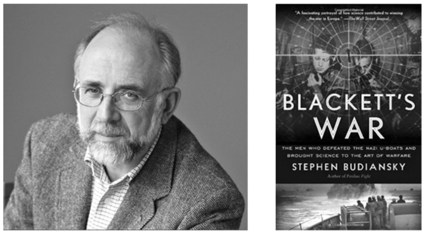


图 18-28　Stephen Budiansky 和 *Blackett's War*

“闪电战”开始时，英国的情报工作控制在类拉丁文和希腊文科目上取得杰出成绩人的手中。然而，这些简单的计数显然不需要具备高等数学和概率论知识，第一次需要它们出场的情况是为英国上空出现的空袭给出相应的对策。由出生于剑桥的科学家 Patrick Blackett 带队，一群来自英国的科学家匆忙组成团队去协助情报工作。

在“不列颠之战”期间，该团队为海岸司令部工作。在那里，他们主要把原始雷达系统收集的数据与实际战争结合起来，设法减少击落一架德军轰炸机所需要的炮弹数量——大约从 20000 发降至 4000 发。

Blackett 的小组及科学家们对在“不列颠之战”中取得重大胜利的措施进行修改，在提供建议之前，他们梳理了操作记录并陪同参加任务，旨在发现有益的模式并收集宝贵的数据。同时，他们参与了德军通信密码的破译——使用最早的 IBM 和 NCR punch-card 计算机进行辅助，而对手则是恩尼格玛密码机。从这些混乱的数据中，Blackett 的小组分析出了至关重要的信息，例如在遭遇德国潜水艇后攻击机需要的挂载时间（15 秒）、深水炸弹的最佳设置（25 英尺）及最佳的爆破模式等。

战争结束 3 年后，Patrick Blackett 被授予诺贝尔物理学奖。

至于前面提到的大西洋护航问题，当时海军与空军的将领们在很长一段时间内都拒绝改变他们的信条，他们认为小编制比大编制更安全（尽管有铁证证明这是错误的）。Blackett 证明了在一个 15～24 艘舰船混编的舰队中，每艘船有 2.3% 的概率被击沉；而在舰队中的船只数量大于 45 艘时，每艘船被击沉的概率只有 1.1%。

这里有一个前提性的假设，那就是每一枚鱼雷击中盟军舰船的概率为 p，每一艘潜艇的载

弹量为 q，这两个值都是有限的。U 型潜艇没有装配雷达（二战时期，只有 1942 年以后才有部分美国潜艇装配雷达，用于在海面航行时观察远处的舰船目标），也就是说，U 型潜艇对附近目标的观测几乎只能靠舰长探出头用望远镜去看，这种观测方式在极为晴好的天气状况下能看到 30 千米以外的舰船，但一般情况下只有 10 多千米的视野。

从美国东海岸的纽约到英国伦敦的直线距离大约是 6000 千米，到英国西海岸的各个港口城市的直线距离也在 5500 到 6000 千米，而航线通常不是完全笔直的，所以实际的航行距离要比这个数字大一些。这个航线在南北宽度上的分布可能超过 500 千米。

在 U 型潜艇的有效观测距离为 10 千米的情况下（也就是一个半径为 10 千米的圆形），6000 千米的直线长度需要至少 300 艘 U 型潜艇才能全面覆盖；而如果按照 500 千米的航线可能宽度来计算，需要 25 艘 U 型潜艇才能完全覆盖。这样算下来，对整个北大西洋航线，需要 7000 艘 U 型潜艇才能把所有的盟军船只抓个“干净”。第二次世界大战中，德国共建造 U 型潜艇 1131 艘，加上战前建造的 57 艘，共 1188 艘。此外，潜艇要在海洋中游弋来捕捉战机，遭遇了就打，打不过就跑，遇到没油、没弹药的情况也要回港补充。所以，这种排布方式只能用来计算数量的上限，没有实战意义。

被派出的德国 U 型潜艇在北大西洋中四处游弋，而北大西洋中来往运输的商船平均每天有 1500 多艘。一艘潜艇要想在这样一个环境中逮到盟军的商船，就好像在一个宽阔的游泳池水面上放一些乒乓球，然后让人闭着眼睛在里面仅凭双手去摸一样。在这种人和球的数量都非常有限的情况下，显然参与摸球的人越多，球被摸到的概率就越大。

起初的护航舰队都是小规模的，甚至是单艘商船从美国东海岸出发前往英国。在任何一刻，如果我们能做一张快照，一艘舰船就相当于游泳池水面上漂浮着的一个乒乓球，而北大西洋就相当于这个游泳池。如果把多个乒乓球捆在一起放在游泳池里，因为球的分布更为集中，人在这一空间里与球相遇的概率变小了，寻找的难度就会增大。也就是说，当船只数量一定的时候，组织规模较大的护航编队肯定比规模较小的护航编队更难被 U 型潜艇遇上。

还有一个问题，就是在潜艇搭载的鱼雷命中率为 p 且载弹量为 q 的时候损失有多大，这也是那些海军高级将领在战争之初所担心的事情——他们总担心大规模的护航编队在被敌人发现后被“团灭”。

大部分 U 型潜艇的鱼雷搭载量为 14 ~ 22 枚[137]。我们假设 U 型潜艇在遭遇护航编队后连续射击，且每发鱼雷的命中率都为 p，接下来就是一个利用伯努利分布的特性求解概率的问题了。

假设发射第 1 枚鱼雷时，命中 1 艘商船（爆炸并击沉）的概率为

$$p$$

137　VIIF 型潜艇可搭载 39 枚鱼雷，但是该型号的潜艇只生产了 4 艘。

命中 0 艘商船的概率为

$$1-p$$

发射第 2 枚鱼雷时，命中 2 艘商船的概率为

$$p^2$$

命中 1 艘商船的概率为

$$2p(1-p)$$

其中，第 1 枚未命中且第 2 枚命中的概率为 $p(1-p)$，第 1 枚命中且第 2 枚未命中的概率也为 $p(1-p)$，这样得到的总命中率为 $2p(1-p)$。

命中 0 艘商船的概率为

$$(1-p)^2$$

发射第 3 枚鱼雷时，命中 3 艘商船的概率为

$$p^3$$

命中 2 艘商船的概率为

$$3p^2(1-p)$$

其中，同时满足第 1 枚命中、第 2 枚命中、第 3 枚未命中的概率为 $p^2(1-p)$；同时满足第 1 枚命中、第 2 枚未命中、第 3 枚命中的概率为 $p^2(1-p)$；同时满足第 1 枚未命中、第 2 枚命中、第 3 枚命中的概率为 $p^2(1-p)$。$3p^2(1-p)$ 由这 3 种情况的概率相加而得。

命中 1 艘商船的概率为

$$3p(1-p)^2$$

其中，同时满足第 1 枚命中、第 2 枚未命中、第 3 枚未命中的概率为 $p(1-p)^2$；同时满足第 1 枚未命中、第 2 枚命中、第 3 枚未命中的概率为 $p(1-p)^2$；同时满足第 1 枚未命中、第 2 枚未命中、第 3 枚命中的概率为 $p(1-p)^2$。$3p(1-p)^2$ 由这 3 种情况的概率相加而得。

命中 0 艘商船的概率为

$$(1-p)^3$$

……

这样一个一个排下去。为了统计方便，我们做一个表格来看一下（如表 18–2 所示）。

表 18–2　5 枚鱼雷遭遇 5 艘商船组成的编队

	5 艘	4 艘	3 艘	2 艘	1 艘	0 艘
1 枚					p	$1-p$
2 枚				p^2	$2p(1-p)$	$(1-p)^2$
3 枚			p^3	$3p^2(1-p)$	$3p(1-p)^2$	$(1-p)^3$
4 枚		p^4	$4p^3(1-p)$	$6p^2(1-p)^2$	$4p(1-p)^3$	$(1-p)^4$
5 枚	p^5	$5p^4(1-p)$	$10p^3(1-p)^2$	$10p^2(1-p)^3$	$5p(1-p)^4$	$(1-p)^5$

展开到 5 层，看得就比较清楚了，纵列表示发射的鱼雷数，横列表示击沉船只的概率，其中的规律也非常明显，系数是一个典型的贾宪三角。如图 18–29 所示，14 枚鱼雷遭遇 14 艘船只（也就是盟军舰船充足）的情况下，不会出现鱼雷没有打完时商船全部沉没的情况。

关于二战时期德国鱼雷的命中率问题，我查了很多相关资料，但没有得到比较可靠的数据，只是在和军事发烧友讨论的过程中听到了一些相对信服人数比较多的数据——大约 30%。在找到更权威的数据之前，将 $p = 30\%$ 代入看看。

可以看到，在命中率为 30% 的前提下，14 艘舰船的编队被搭载 14 枚鱼雷的 U 型潜艇“团灭”的概率只有 4.78×10^{-8}，这是一个几乎可以视为 0 的数字了。而平均每次（数学期望）将 14 枚鱼雷射出，击沉的舰船数量仅为 1.48 艘（这与 Blackett 的估算值有差异，产生差异的原因应该是 p 的取值不同），概率最大的情况是击沉 7 艘舰船，只占 7% 左右。从数学计算上来看，这种情形已经非常理想了。如果 14 艘舰船分成 14 个单独的编队，那么每条船被击沉的概率可是要乘以 14 的。

即使击沉概率 p 上升到 50%（对于鱼雷来说这个命中率已经非常高了），整个方案也会输出一个不太坏的值（如图 18–30 所示），“团灭”的概率只有 6.10×10^{-5}，每次（数学期望）将 14 枚鱼雷射出，击沉的盟军舰船的平均数量也不过为 3.75 艘。

而我们刚刚讨论过，大编队的好处是让 U 型潜艇碰到盟军舰船的概率降低。理论上，编队越大，“收益”越好。在刚刚这个例子中，14 艘舰船组成一个编队被发现的概率是 14 艘舰船每艘单独编队被发现的概率的 $\frac{1}{14}$（也就是原来被发现概率的大约 7.1%）。

在采纳这项建议后，盟军的很多商船先在飞机和军舰能够保护的近海范围内集结，集体通过相对危险的海域，在到达距离不列颠岛比较近的海域后分散开来，向各自的目标港口驶去，盟军舰队遭袭被击沉的概率由原来的 25% 下降为 1%，大大降低了损失。

1943 年以后，盟军的飞机和护航舰船的反潜能力由于声呐和雷达技术的改进而获得了很大提升，大量的飞机和军舰也投入到护航和反潜的任务中，盟军在北大西洋的舰船损失降低了 65%，北大西洋上的盟军优势完全不可逆转。而到了 1944 年，只有 200 艘舰船被击沉。第二次世界大战期间，德国的 1188 艘潜艇击沉了 3500 艘舰船，造成 45000 人死亡。到战争结束时，有 781 艘德国潜艇被盟军击沉。

横：艘 纵：枚	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Exp
1														0.3	0.7	0.3
2													0.09	0.42	0.49	0.6
3												0.027	0.189	0.441	0.343	0.9
4											0.0081	0.0756	0.1764	0.4116	0.2401	1.0236
5										0.00243	0.02835	0.1323	0.3087	0.7203	0.16807	1.86015
6									0.000729						0.117649	
7								0.0002187							0.082354	
8							0.00006561								0.057648	
9						0.000019683									0.040354	
10					5.905E-06										0.028248	
11				1.77147E-06											0.019773	
12			5.31441E-07												0.013841	
13		1.59432E-07													0.009689	
14	4.78297E-08	1.45083E-06	2.03117E-05	0.000173778	0.0010137	0.004257553	0.01324572	0.0309067	0.054087	0.070112	0.065438	0.041643	0.016194	0.016194	0.006782	1.483288

图 18-29　命中率为 30% 的情况

横：艘 纵：枚	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Exp
1														0.5	0.5	0.5
2													0.25	0.5	0.25	1
3												0.125	0.375	0.375	0.125	1.5
4											0.0625	0.25	0.25	0.25	0.0625	1.75
5										0.03125	0.15625	0.3125	0.3125	0.3125	0.03125	2.65625
6									0.015625						0.015625	
7								0.0078125							0.007813	
8							0.00390625								0.003906	
9						0.001953125									0.001953	
10					0.0009766										0.000977	
11				0.000488281											0.000488	
12			0.000244141												0.000244	
13		0.00012207													0.000122	
14	6.10352E-05	0.000793457	0.004760742	0.017456055	0.0436401	0.078552246	0.104736328	0.1047363	0.078552	0.04364	0.017456	0.004761	0.000793	0.000793	6.1E-05	3.750732

图 18-30　命中率为 50% 的情况

18.2.3　数学家的天下

英军船队在大西洋航行时，经常受到德军潜艇的攻击。因为深水炸弹是根据定深来引爆的，而不知道潜艇的具体下潜深度会使得定深非常不准，进而导致中率大大降低，因此英国空军的轰炸对潜艇几乎不构成威胁。为解决这一问题，英军请来一些数学家进行专门研究，发现潜艇从发现英军飞机开始下潜到深水炸弹爆炸时只下潜了 7.6 米，而在这段时间里炸弹往往已经下沉到水面以下 21 米处。经过科学的论证，英军尝试调整深水炸弹的定深，使爆炸深度从水下 21 米减为水下 9.1 米，攻击效果较过去提高了 4 倍，这也使德军误以为英军发明了新式炸弹。

此外，空军方面也有一些通过战损分析进行战术改进的事例。

当德国对法国等几个国家发动攻势时，英国首相丘吉尔应法国的请求，动用了十几个防空中队和德军作战。这些防空中队必须由大陆上的机场来维护和操作。在与德军的空战中，英军飞机损失惨重。与此同时，法国总理请求英国增派 10 个防空中队，丘吉尔决定同意这一请求。英国内阁知道此事后，找来数学家进行分析预测，并根据出动飞机与战损飞机的统计数据建立了回归预测模型。研究发现，如果补充率和损失率不变，飞机数量的下降是非常快的，用一句话概括就是“以现在的损失率损失两周，英国在法国的‘飓风’式战斗机便一架也不存在了”，于是要求内阁否决这一决定。最终，丘吉尔同意了这一要求，并下令让除了留在法国的 3 个中队以外的飞机全部返回英国，为下一步的“英伦保卫战”保留了实力。

18.2.4　小结

在这一节我们看到了一些真实的例子——用数学与数字拯救大量战士和百姓生命的例子。数字中确实蕴含着极为丰富的信息，只要能够合理解读它，就会激发其中蕴含的巨大能量。这还不够吸引人吗？

大数据在近几年越来越火。我坚信，其中过度炒作的成分是少的，而未来在行业市场成熟后从其中获取巨大利益的需求才是真正的市场性需求。诚然，在数据科学本可以展现魅力的大数据市场，目前还存在很多问题，不论是基础建设还是人才的匮乏，都是比较尖锐的矛盾。不过，这不能阻止人们向这个理想中的目标奋勇攀登。爱上数据科学，以数据科学的思维方式来对待一切工作，我想这一天的到来真的不会太远了。

18.3　人人都能运筹帷幄

在这里，我需要对本书的内容做一个简明扼要的归纳。

从内容上看，本书涵盖的内容非常多。“认知篇”中谈到了有关认知和辩证哲学的问题，总结起来只有一句话——量化认知，实验为本，辩证思维是科学认知的根本要点。“分化篇”对几

个与数据科学关系相对紧密的分支学科进行了介绍。实际上，数据科学涉及内容几乎涵盖所有对观测数据进行科学处理的内容，是一个非常庞杂的科学体系。统计学是侧重对大量数据的观察和认知的学科，信息论是围绕信息量进行研究的学科，混沌论体现的是对系统中不确定性现象的研究方法，算法学则讨论如何合理利用计算机的资源调度方式去解决一个问题模型，这些都是分支学科对数据科学支持最为重要的部分。“实践篇”对数据生命周期的每一个细节应该考虑的内容都进行了完整的讨论。应该说，每个部分在数据生命周期中都是有价值的。

数据科学和其他科学体系下的学科一样，都有完备的认知体系。精确性、体系性是一门学科是否可以被认定为科学的重要标志。数据科学的分支都在各自深入研究的方向为我们研究客观世界提供了帮助——那就是熵减。从事物之间的关系无法量化，到定性，到量化，再到精确量化，这个熵减的过程就是数据科学一直在研究的过程，也是一个精益求精的过程。人类必将迈着坚实的步伐，向数据科学光辉的未来前进。

第 19 章　本书相关内容问与答

问：数据科学现在这么火，有没有学习的路线图呢？

答：数据科学是一种研究信息感知、抽象、保存、建模、传输，以及数据之间逻辑、数量统计与计算和转化关系的应用科学。数据科学的本质就是表述和指导对事物认知的关系量化，把普适性的科学思维方式应用到数据上，使其成为一门窄而深的、体系完整的、精确的学科。这就是数据科学要解决的本质问题。也就是说，只要是做其中一个方面或者一个子领域的工作，就是在研究数据科学。只不过我们平时口中的“数据科学”通常偏重于数据分析，即通过对数据的挖掘与分析消除不确定性，汲取剩余价值。单纯从这个角度来说，我认为应该学习的领域有这样 4 个。

（1）统计学

通过对统计学的系统学习，熟悉大量样本分析过程中的注意事项和技巧，以及科学的分析方法论。

（2）高等数学

通过对微积分领域的学习，理解一种哲学或思维方式。

（3）数据挖掘和机器学习

将这两个领域一起研究，而且不建议强行区分两者。这两个领域都是在研究数据内在联系的逻辑，并寻求用简洁的方式加以描述和印证。

（4）深度学习与人工智能

这个领域在未来是非常有前途的，因为“用机器代替人”本身就是一个历史趋势，而且从未改变。所以，这个领域是一个经得起时间考验的研究领域，在未来几十年都会有很好的应用场景，而且随着计算机处理能力的增强，这个趋势也会越来越明显。

问：数据科学和大数据有什么区别？它们是同一种东西吗？

答：数据科学是一个笼统的概念，而且每个人对这个概念的认知与定义都不相同。我建议不要太过纠结这个概念圈定的范围是什么，只要把一种科学的认知观念带到日常工作和实践中就可以了。

大数据比数据科学的范围更为具象。数据通过各种软件进行收集，通过网络进行传输，通过云数据中心进行存储，通过数据科学家或者行业专家进行建模和加工。通过数据分析得到的是一种知识，是一种人们通过数据洞悉世界的能力。数据之间原本错综复杂的潜在关系，会使大量孤立且多来源的数据因为同时出现在一个舞台而显得更为有趣，大量看似不相关的事情却能够通过观察与分析使人们知晓更多背后的因果。这些因果联系的意义会让人们能够在各个方面推测未来的趋势，减少试错的机会，缩减成本，降低风险，解放劳动力。这是大数据产业本身的价值与意义所在。

问：数据科学能解决所有的问题吗？

答：数据科学作为一种贯穿于所有生产与认知领域的学科，是一种类似方法论的学科。因为未知事物永远比已知事物多，所以如何正确地、科学地认知未知事物并使其成为已知，就是一个指导思想方面的问题了。形成这样一种思维方式比具体解开一个局部性的问题更为重要。

数据科学的套路也非常固定。

（1）建立假说

建立一个概念假说作为研究的目标。

（2）观测与证明

将这个假说的条件和环境设定放到生产生活中加以观察，或者干脆在实验室环境中设计一个完整的实验，通过不断地观测，尤其是量化观测，对假说的内容进行证明或证伪。因为数据科学本身就注重精确性（也就是量化环节），而且一切科学全都是以实验为本，也就是说，以观测到的结果作为认知依据，以此判定是否应验，所以这个量化观测最为关键。

（3）归纳与调整

在观测的过程中会获得大量的观测数据，要对这些数据进行充分且科学化的分析，然后根据这些数据进行归纳总结，进而对假说的内容进行相应的调整。因为最初假说中的内容很可能与实际的观测结果有出入，所以这个时候就要在辩证思维的帮助下去发现和排除观测中不周密的因素，或者重新审视假说的严谨性，并进行相应的调整。

这种套路可以复用在很多的科学与生产场景中，屡试不爽。

问：数学不好是不是就不能学习数据科学呢？

答：首先，数据科学作为一种贯穿于所有生产与认知领域的学科，是一种方法论类型的学科。既然是方法论，那么学习过程中要着重理解其运作方式。

所谓数学不好，或许是指无法在课堂上解决数学题或者不具备用数学建模的方式解决复杂问题的能力。但是，对于事物的认知，尤其是基于统计的认知，关键在于悉心观察与思考，而

是否一定要用高深的数学功底去解决一个问题，要视所在领域而定。所以，数学不好对于学习数据科学的影响是有限的，我们只要能在自己擅长的领域去应用数据科学的认知方式就好。

问：数据科学能帮我们解决具体问题吗?

答：数据科学是一种方法论类别的学科。方法论类别的学科最大的劣势就是与解决具体问题的距离相对比较远。在日常生产和运营场景中，除了一般性的认知问题以外，还有决策问题等可能也需要我们用数据科学的思路去解决。

在日常生产和运营中，几乎避免不了要在决策中对多个方案进行比较和裁决。但是，比较和裁决通常会由于没有相应的标准而争执不下，所有人都会说某种方案好或不好、有利或不利。如果只是到此为止，那么大家就都是靠发表感慨的强烈程度或者声音大小来做判断了，这是难以服众的。

如果要用数据科学的观点来解决这类问题，通常有这样两个步骤。

（1）明确目标或目的

决策的内容通常是一个方案是否要去实施，或者在多个方案中哪个好、优先做哪个的问题。那么，裁定的过程就需要明确究竟以什么目的为准，是收益相等的情况下投入最小，投入相等的情况下收益最大，还是收益比投入的商值最大。这 3 种方案都是有道理的，只是收益与投入的核算边界要清晰，要明确规定是只计算材料成本，还是要同时计算人员及其他管理成本等。

（2）量化

决策的实质是一个评价过程或比较过程。只有有大小的东西才能比较，也就是说，决策中的每个方案、每个提议都要能够量化才具有说服力。即使争执再多，再赞同自己的观点或不赞同对方的观点，都要基于具体的数据说话。

数据这东西有一个奇怪的作用——即使人们再质疑其准确性，也是有比没有要好，权威数据比民间数据要好。有了数据，就可以把问题量化、简化。所以，如果不想糊里糊涂地做决策，那就用翔实的数据来做自己的武器吧。

问：数据科学研究问题的时候应该怎么去划定一个问题的边界呢?“打破砂锅问到底”这样的精神应该贯彻到什么程度呢?

答：对一个研究对象，首先要划定研究的边界。我认为，这个边界在具体的生产中可以只限定于解决一个或一类基本问题（痛点）即可。否则，无休止地“打破砂锅”也会大量消耗成本。

以本书开头提到的历法研究问题为例。在这个问题中，核心研究的对象是地球公转周期、自转周期及附带的气候变化问题，所以把带有这几个因素的模型讨论清楚就可以了。这个问题属于初等天文学和地理学研究的范畴，我们只要掌握天体运行周期之间的关系就足够了，没必要

深究。至于导致四季变化的太阳直射角度问题，即为什么太阳直射会比斜射使地面温度更高，往深里研究是物理学和原子学的范畴，可以在其他学科里单独研究和讨论，而如果把这个问题也放在历法研究中的话，就会使历法的研究变得过于复杂，反而不利于讨论。所以，边界的划定在一个问题的研究中还是非常重要的。

问：我知道数学应该是挺有用的，但是为什么我对学习数学一点兴趣都没有？有什么方法可以帮助我学习数学吗？

答：数学是一种抽象科学，是把很多事物中共通的模型抽取出来进行量化研究的学科，它的特点有如下两个。

（1）抽象

数学作为已经剥离得一点“血肉”都没有的“骨架”，看上去当然不会让人有亲近感。但正因为数学把这些无关的维度从人类的视线感知中移除，才使人类更能把精力集中在对共通性质的研究上。

（2）泛化性好

所谓泛化性好，就是可以进行迁移应用，即换到任何一个场景中，只要模型一致或者类似，就可以应用。这个应用就是一个认知维度最小范围扩大化的过程，所以，在这个过程中我们会明白为什么以前在学校里总听老师说“要理论联系实际”。理论是认知维度极小化的精简描述，是忽略其他因素的割裂性的描述，其本质是认知维度扩大的迁移过程，因此有差异是理所当然的事情。

如果对数学没兴趣，可以把数学套用在你喜欢的场景里去学习。例如，《九章算术》中每个问题的场景都是实际生产中可能发生的，所以民间传抄就非常踊跃，根本不用人去鼓励。换成我们自己，可以套用经济学里的理财收益计算、理财产品设计思路，计算机领域的服务器负载计算与预测，以及身边任何我们感兴趣或者与我们的利益相关的场景，估计会有比较好的效果。

问：有没有必要研究 Hive、SparkSQL、MySQL 这些开源数据库软件的源代码呢？

答：这个问题要看个人的职业规划如何。如果你现在供职或者未来计划供职的是一家对这些软件有很强的个性化需求（例如对性能要求很高或者对数据支持规模很大）的公司，并且愿意投入足够多的人力来进行代码的修改和维护的话，还是可以花一些力气去研究的。否则，就个人修改这些代码并提交到 GitHub 而言，通常会由于你的修改是一种偏重场景的修改而导致受众较少，这意味着可能没人愿意与你合并代码，也就没有人愿意帮你一起维护这个 Branch，你要永久独自维护这个 Branch，花费的时间成本非常高。不过从个人兴趣和学习的角度来看，对这些开源软件的代码进行研究肯定是没有坏处的。

问：既然数据科学这么好，数字化这么好，为什么现在仍然有很多地方没有进行数字化呢？

答：数字化可以解决描述的简洁性和精确性问题，对完整的体系性、精确性认知有极大的帮助。然而，在数据科学和数字化落地的过程中，我们不能不注意落地本身的成本问题，因为这涉及这种行为是不是可以市场化，以及在市场化之后是不是能够得到广泛的认可。

现在大家看到的数字化落实比较好的领域，大都是高、精、尖领域及体量比较大的产业领域，因此在一些个人、小型企业和个体工商单位做数字化落地通常是很困难的，主要原因就是数字化落地的成本与收益的问题。在高、精、尖领域和大体量产业，投入的数字化和数据分析成本可以通过其体量的红利来产生大量的剩余价值，因此在这些领域进行数字化的阻力当然比较小，数字化甚至会成为其顺利进行的不可或缺的工具。而对于个人、小型企业和个体工商单位，数据科学及数字化的落地就意味着要投入成本，而这个成本在当前的体量下无法获得好的加成效果，因此对其剩余价值也就很难有所期许。

所以，数据科学或数字化的落地与研究是否适合当前的个体或组织的需要，仍然是一个 ROI 的决策问题。我认为，如果要在全社会范围内实现数据科学的普及化和数字化，还是需要走服务化的道路，提供高内聚、低耦合的数据服务才是普及的最佳路径。

问：数据科学如何帮助公司的日常管理？有方法可循吗？

答：数据科学作为一种认知科学，其本身有相对固定的实施套路。公司的日常管理，说到底是协作问题。好的管理会使协作效率提高，不好的管理会使协作效率降低。

如果希望从数据层面对日常管理的有效性进行改进，应该注意日常工作效率的指标化，并不断通过各种局部的战术层面的工作方式改进与已有的工作模式进行对照，用这种方式来量化解释新的工作方式是否有助于协作效率的提高。这种方法很有效，而且极有说服力。如果你是一位希望提高工作中协作效率的管理者，就应该想办法从这一刻开始努力量化工作中诸多对工作效率进行描述的环节，并通过对方法的改进，逐步观察这些效率是否因你的改进而得到了明显的提高。

问：可以用数据科学精确地解释市场经济行为吗？

答：我想问这个问题的目的应该是希望通过对市场数据的分析进行市场预测。

市场是由大量商品供需的成交而形成的体系。市场中的交易动机，不论是为了投机，还是作为最终用户直接使用，在市场上体现出来的都是对某种商品买入或者卖出的情绪——不管是汽车、房子、农产品、石油，还是智能手机、中草药，甚至是某种虚拟或者实体的货币。

影响市场的因素非常复杂。人们购买一些东西，有的时候可能是受周围人的购入或卖出情绪的影响，有的时候可能是受新闻中释放出来的信号的影响。在华尔街有一句谚语：“市场是在

爬一堵担忧的墙。”这句话主要应用在金融市场，大意是说，市场上都是由于某些担忧的情绪，才会去抛出一些风险高的产品，买入一些可以避险的产品。这种对于产品喜爱或厌恶的情绪通常体现在其他产品的供需上，产品价格的波动通常而且一定是由供需不平衡造成的。

市场情绪和其他我们能够感知的内容之间应该有着某种“客观存在”的联系，而大多数情况下的主要矛盾是我们对这些消息的获取能力很低，对相关维度的感知能力很低——无法获取这些数据，造成我们对市场行为的判断不准确。这也是没有办法的事情，因为建立模型的任务确实复杂到了一个人或者一个小型组织无法负担的程度——不知道拉普拉斯妖有没有这个能力，或许有。

当然，在一些情况下，市场对于某些商品的追逐确定性是比较高的。

我们以避险货币日元为例，由于其流动性好，所以在市场上出现一些风险事件的时候，日元往往会表现出大涨的现象。例如，“英国脱欧”、“苏格兰脱英公投”这种风险事件的消息一旦放出，短时间内会就看到日元有一波比较明显的上涨。

除了货币以外，其他商品同样拥有一些影响权重极大的信息维度可以让我们在不参考太多其他因素的情况下判断涨跌，例如房价问题。房价在中国也是一个非常复杂的问题，虽然在很多城市都有一些政策性的行为参与楼市调控，让人感觉预测起来非常困难，总是不知道“泡沫”有多大、哪里是拐点，但是房价的本质仍然是房屋的供需问题。一个城市的房价是上涨还是下跌，主要是当地外来人口的涌入速度和房屋建设速度较量的结果。三四线城市房价上涨乏力的原因是房屋建设太多，没有足够的外来人口支撑买盘压力。因为到三四线城市买房的人可能大都是周边乡村的人，如果这些人的就业问题无法在这些城市得到解决，房价的保持就根本无从谈起。而像北、上、广、深等大城市，只要外来人口持续快速涌入，而房屋的供应无法跟上，那么结果确实只有一个——房价一定有极大的上涨压力。而不知道“泡沫”有多少、哪里是拐点，原因无非就是没有掌握相关数据。一旦能够拿到相关数据，我们就可以用数据认知的方法来处理——“拟合”一下就可以对中短期的房价趋势作出相对准确的判断。至于长期判断应当如何做，真的不好说——谁让这个混沌的世界会不时地给我们带来一些意外呢？

后　　记

这本书写到这里，就要和大家说再见了。掩卷长思，区区数万言也只能略略吹开数据科学的皮毛而已，距离揭示整个数据科学的方方面面还差得很远。如果在和你谈天说地的过程中，哪怕只有几个章节对你的工作有一丝提示，哪怕只有几句话能引起你对数据科学的共鸣，这本书都没有白写。这就是我作为一个在 IT 行业工作了十几年的“老兵”的心里话。

不管你是刚刚进入 IT 行业的新手程序员，还是在业内摸爬滚打多年的架构师，或者是转行不久的数据分析师，我想你在工作中慢慢会发现，所有 IT 职种之间的差别并不大，我们做的是同一件事——利用一切可以利用的软/硬件资源，用性价比最高的方式服务于我们所供职的公司和广大用户。这条路往前看不到尽头，往后也看不到尽头。

“路漫漫其修远兮，吾将上下而求索。”[138]这本书只是帮助各位读者做一些入门的准备，真正的修行才刚刚开始，我们需要解决的问题还有很多。记得我曾经跟要好的朋友开玩笑说：“趁着年轻，我们应该多动动脑、多动动手，难道你不想在变成一把炉灰之前的这 50 多年里多做一些不让自己后悔的事情吗？”

不要辜负我们的黄金时代。我也坚信，黄金时代一定不会辜负那些爱智求真的人！

138　出自屈原《离骚》。

附录 A

A.1　VMware Workstation 的安装

A.1.1　VMware 简介

VMware 是由美国著名的虚拟化产品提供商 VMware 提供的虚拟化软件产品。它基于图形化管理，使用非常简单，对虚拟机指令的执行效率也非常高。

VMware 提供了多个版本以应用于不同的场景。我们通常使用的是 VMware 工作站（VMware Workstation），这个版本主要用于个人计算机的虚拟机搭建。

安装 VMware Workstation 之后，我们可以在不破坏当前 Windows 环境的情况下使用资源隔离的方法再安装一个类似沙箱环境的 Windows 或 Linux 系统。

A.1.2　安装准备工作

1. 硬件配置需求

在安装 VMware 之前，用户需要准备足够的磁盘空间和运行时所占用的内存。为了保证大数据软件流畅运行，请至少保证达到如下硬件配置。

（1）CPU：Intel Core i3。

（2）内存：4GB。

（3）硬盘空间：50GB。

2. 下载

我们可以从 VMware 的官方网站下载最新版本的软件。VMware Workstation 12 的下载地址为 http://www.vmware.com/products/workstation/workstation-evaluation。如果读者使用 Windows 操作系统，可以点击“Download Now”按钮下载，如图 A-1 所示。VMware Workstation 为商业软件，用户可以免费试用 30 天。

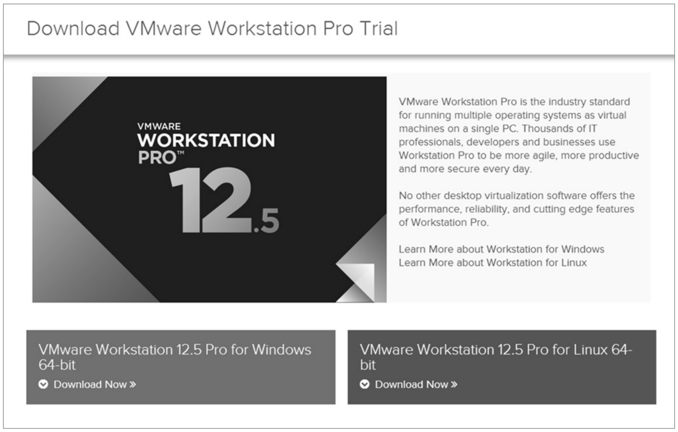


图 A–1　下载页面

3. 安装

下载完成后，进入安装环节。双击安装文件，会进入安装向导界面，单击“下一步”按钮，如图 A–2 所示。

进入 VMware 最终用户许可协议界面，勾选“我接受许可协议中的条款”复选框。单击“下一步”按钮，如图 A–3 所示。

图 A–2　安装 VMware Workstation（1）

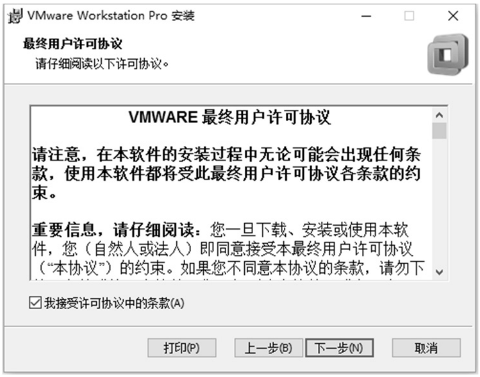


图 A–3　安装 VMware Workstation（2）

接下来，选择默认安装的位置。由于虚拟机的位置可以和虚拟机软件 VMware 的安装位置不同，所以我们可以直接将软件安装在默认的 C 盘上。单击“下一步”按钮，如图 A–4 所示。

建议不勾选“启动时检查产品更新”和“帮助完善 VMware Workstation Pro”复选框。单击“下一步”按钮，如图 A–5 所示。

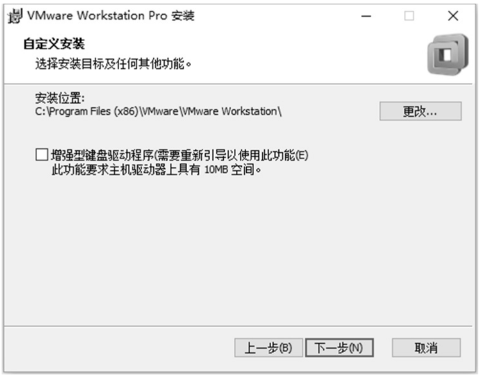


图 A–4　安装 VMware Workstation（3）

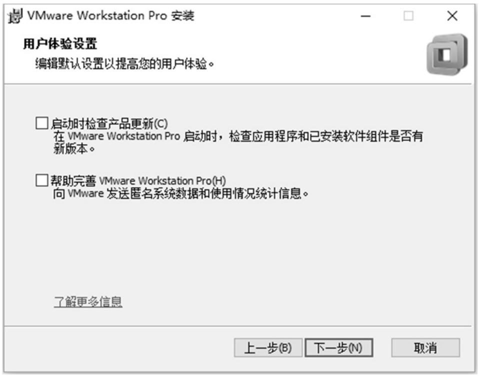


图 A–5　安装 VMware Workstation（4）

为了使用方便，可以勾选“桌面”复选框，在计算机桌面创建快捷方式。单击“下一步”按钮，如图 A–6 所示。

单击“安装”按钮，如图 A–7 所示。等待进度即可，如图 A–8 所示。

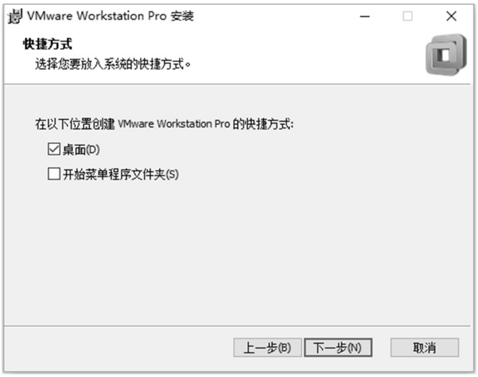


图 A–6　安装 VMware Workstation（5）

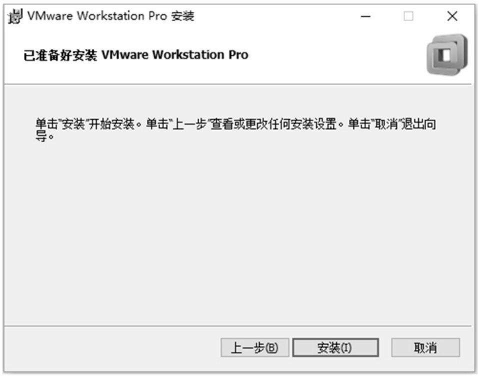


图 A–7　安装 VMware Workstation（6）

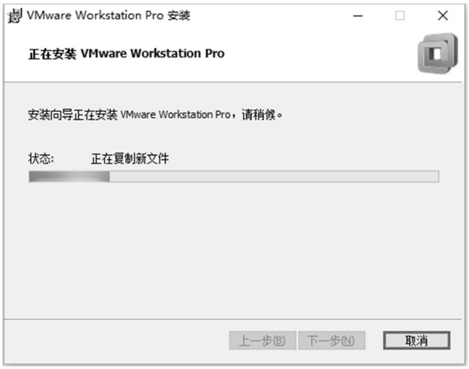


图 A–8　安装 VMware Workstation（7）

A.2　在 VMware 上安装 CentOS

A.2.1　下载

访问 CentOS 官方网站 http://www.entos.org，下载最新版本的 CentOS 系统安装光盘镜像。本书中使用的版本是 CentOS 7，下载地址为 http://isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-1511.iso

A.2.2　安装

启动 VMware Workstation，在初始界面单击“创建新的虚拟机”按钮，如图 A–9 所示。

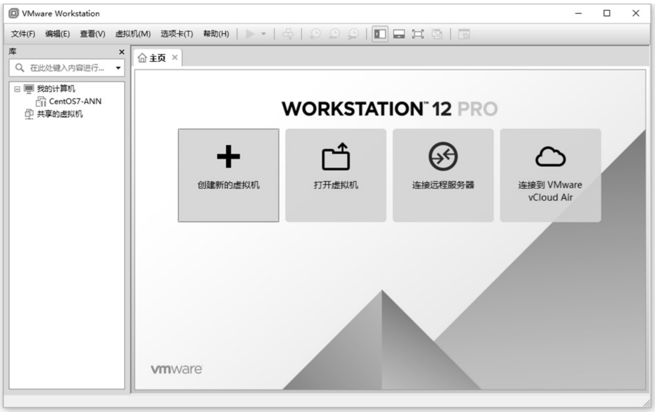


图 A–9　安装 CentOS（1）

选中“典型（推荐）”单选按钮，然后单击“下一步”按钮，如图 A–10 所示。

图 A–10　安装 CentOS（2）

选中“安装程序光盘映像文件(iso)”单选按钮，在下面的设置框中将内容指定为我们刚刚下载的 CentOS 7 系统安装光盘镜像文件的地址，如图 A-11 所示。

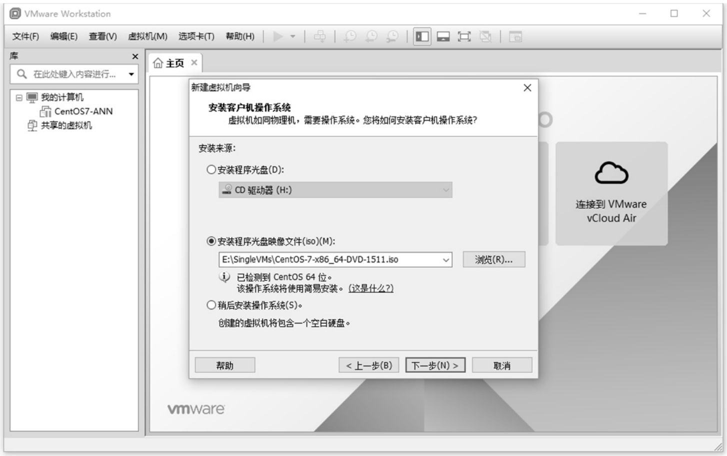


图 A-11　安装 CentOS（3）

这个版本的 VMware Workstation 支持简易安装模式。为了讲解方便，这里把 Linux 虚拟机的全名、用户名、密码都指定为“data”，如图 A-12 所示。

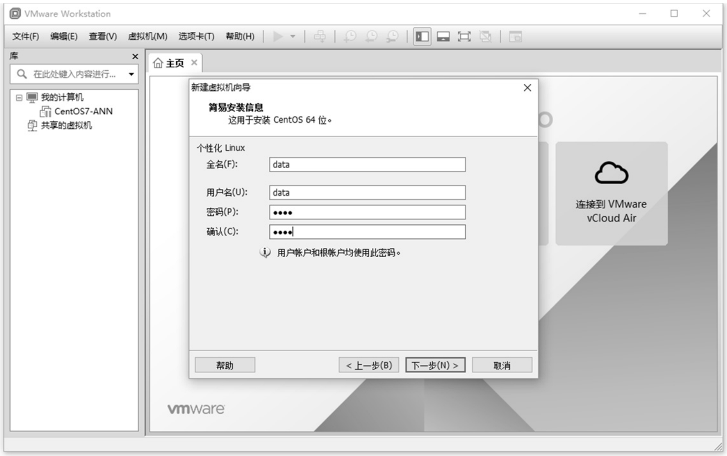


图 A-12　安装 CentOS（4）

指定虚拟机的名称（这是在 VMware 管理中使用的名称，不是其主机名）和安装位置，如图 A-13 所示。

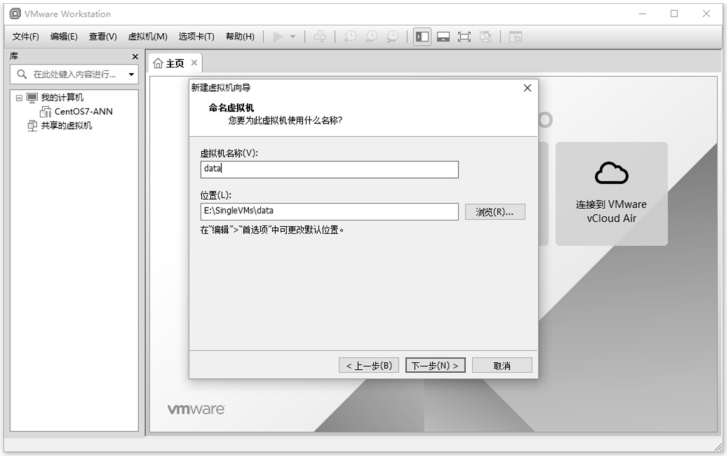


图 A–13　安装 CentOS（5）

给虚拟机指定一个磁盘大小。这里设置的是 100GB，如图 A–14 所示。虚拟机会用按需分配的方式进行动态扩展，所以我们不用在一开始就因为自己的磁盘空间不够 100GB 而担心程序报错。

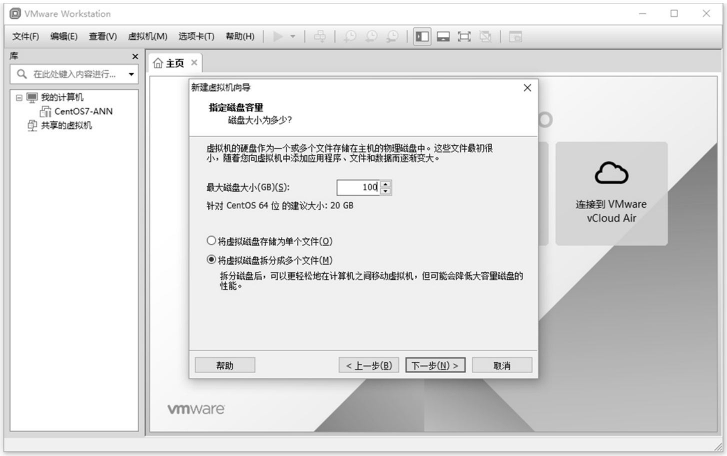


图 A–14　安装 CentOS（6）

单击“完成”按钮，引导虚拟机启动，如图 A–15 所示。

为了加快安装速度和实验速度，建议在硬件资源充足的情况下赋予虚拟机更多的硬件资源，在这里设置了 4 个 CPU 内核和 8GB 内存。通常情况下，不要把超过 50% 的内存和 CPU 资源划分给虚拟机。

图 A-15　安装 CentOS（7）

在简易安装模式下，Linux 会自动进行引导，如图 A-16 所示。之后就等待 Linux 一步一步自动安装，不需要人为干预。

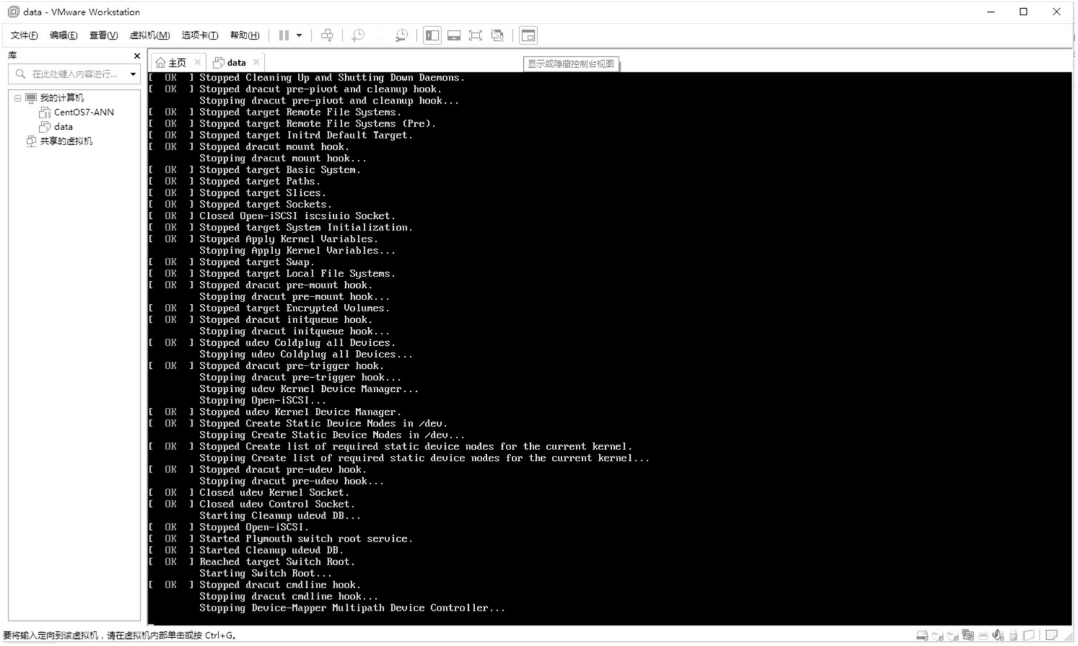


图 A-16　安装 CentOS（8）

单击"Not listed?"选项，使用 root 用户模式，输入密码"data"，这样在实验环境中会更方便一些，如图 A-17 和图 A-18 所示。

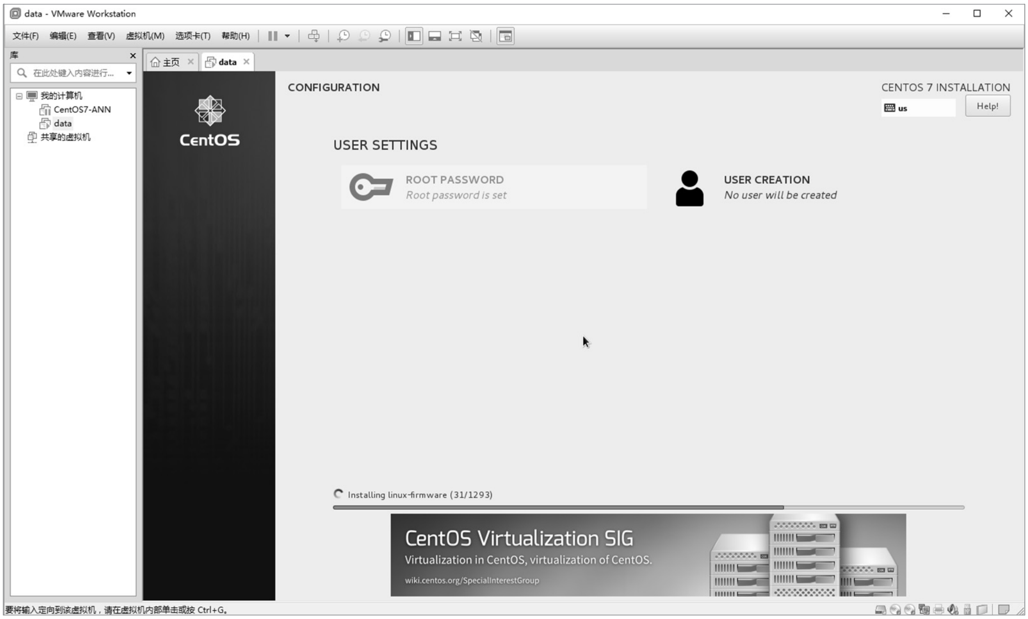


图 A-17 安装 CentOS（9）

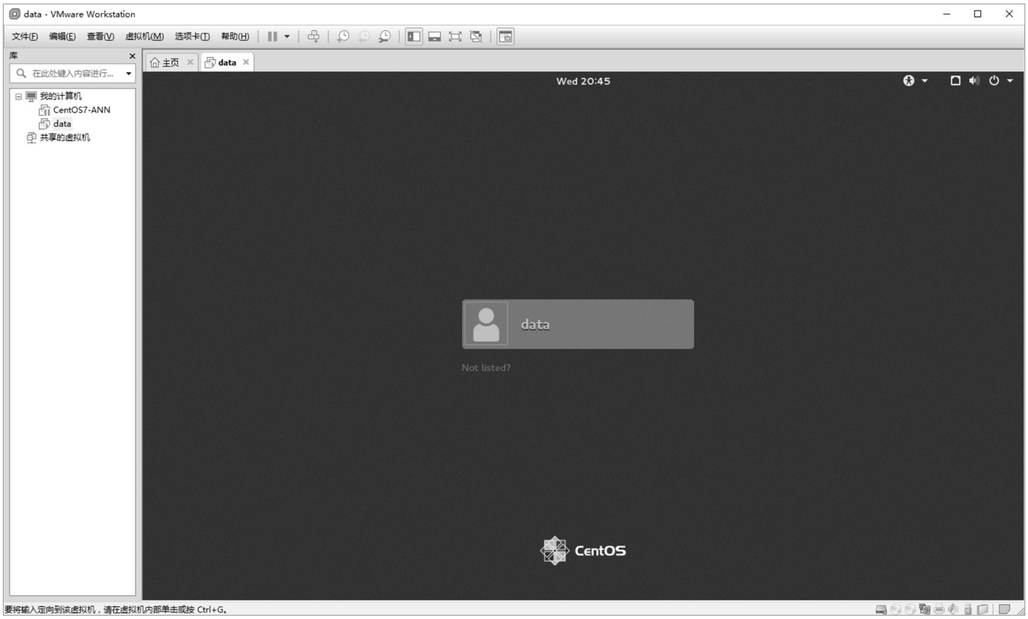


图 A-18 安装 CentOS（10）

进行 GNOME 桌面的初始化设置后，单击“Next”按钮，如图 A-19 所示。

单击“Start using CentOS Linux”按钮，完成安装，如图 A-20 所示。

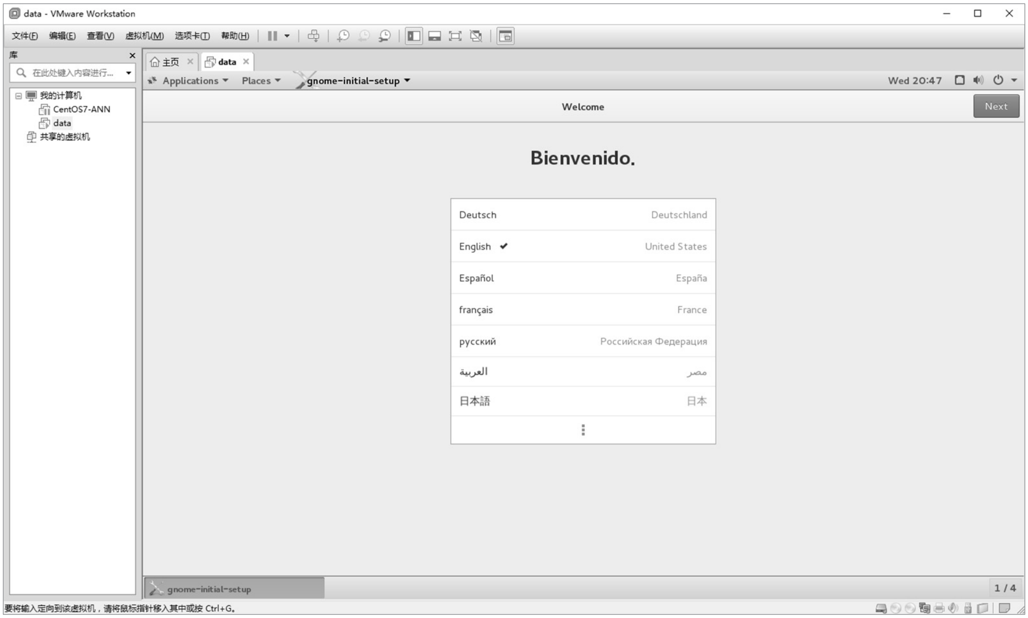


图 A-19 安装 CentOS（11）

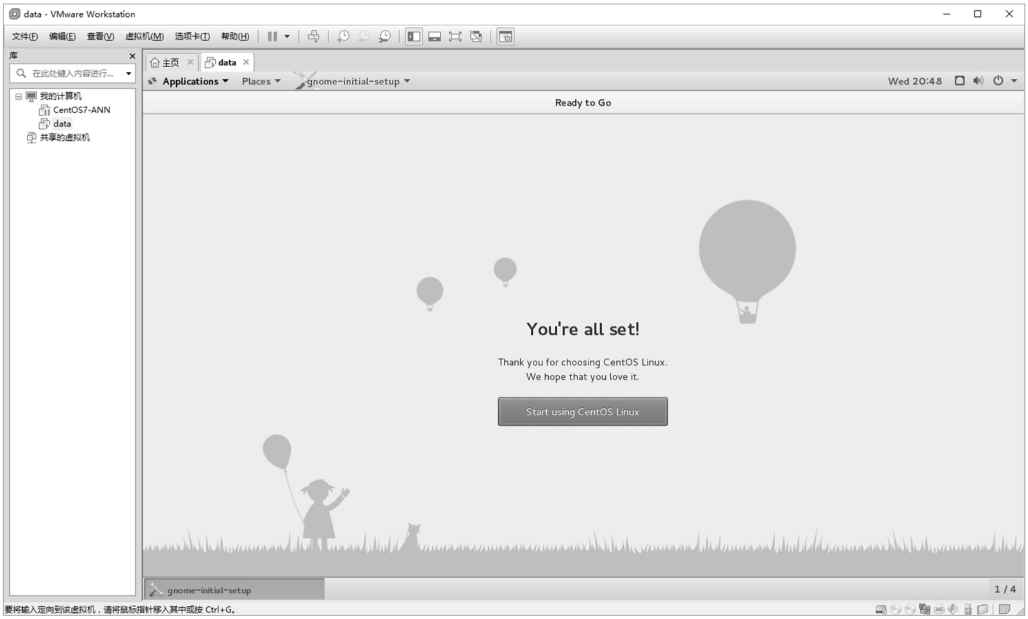


图 A-20 安装 CentOS（12）

现在，我们已经可以使用 CentOS 7 的 GNOME 桌面了，如图 A-21 所示。

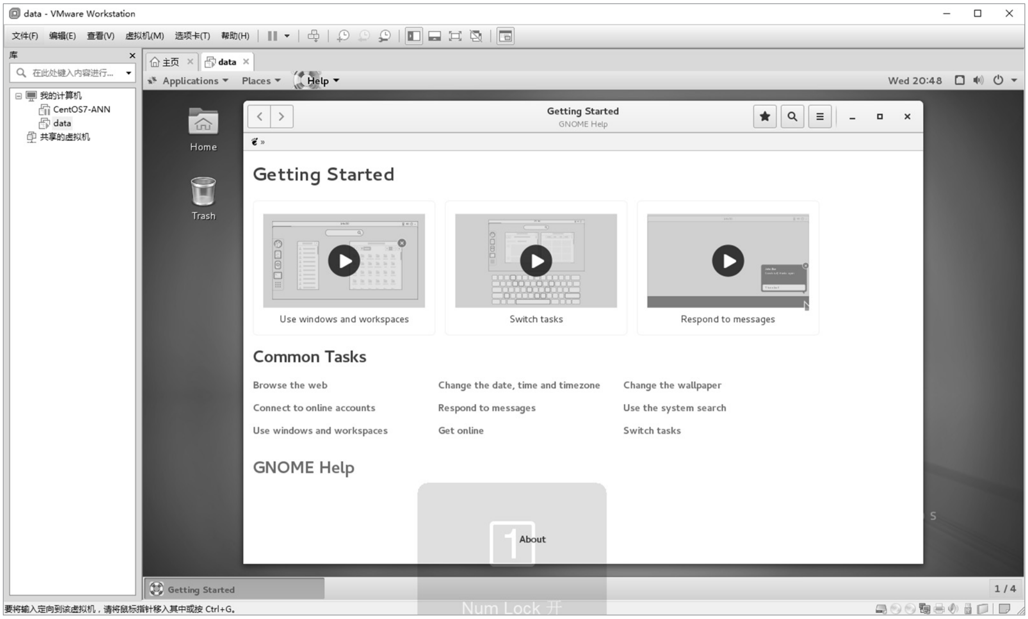


图 A-21 安装 CentOS（13）

A.3 在 VMware 上安装 Ubuntu

启动 VMware Workstation，在其初始界面单击“创建新的虚拟机”按钮，如图 A-22 所示。

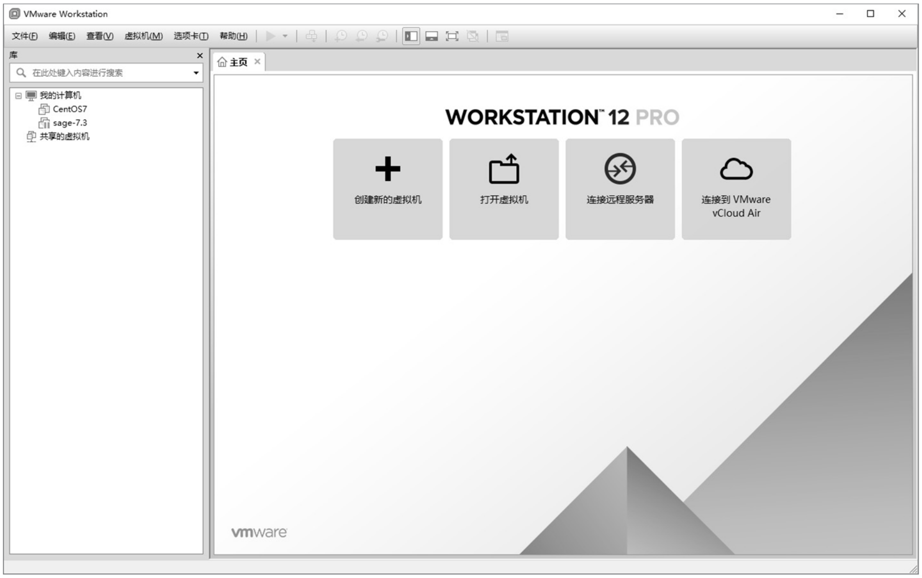


图 A-22 安装 Ubuntu（1）

在这里直接选中“典型(推荐)”单选按钮，然后单击“下一步”按钮，如图 A-23 所示。

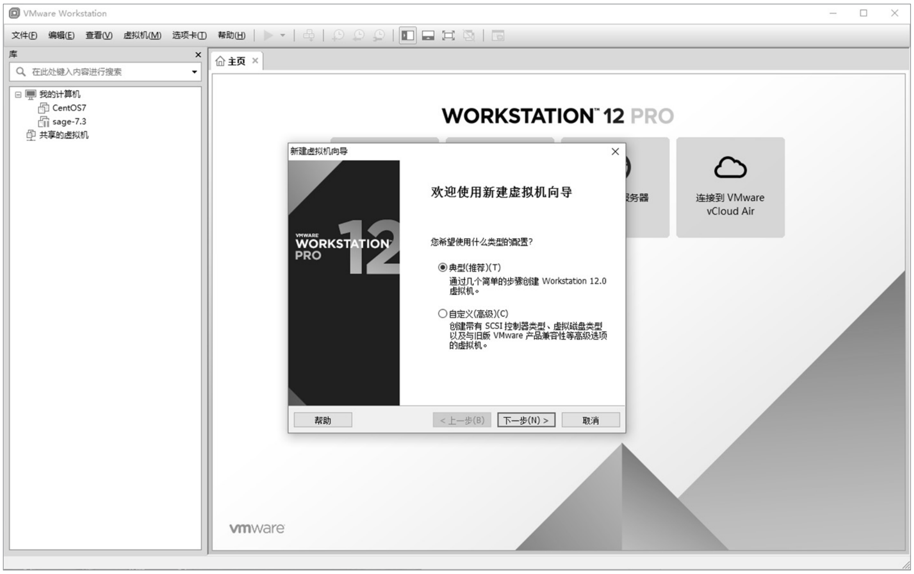


图 A-23 安装 Ubuntu（2）

选中“安装程序光盘映像文件(iso)”单选按钮，在下面的设置框中将内容指定为 Ubuntu 16 安装文件的地址，如图 A-24 所示。

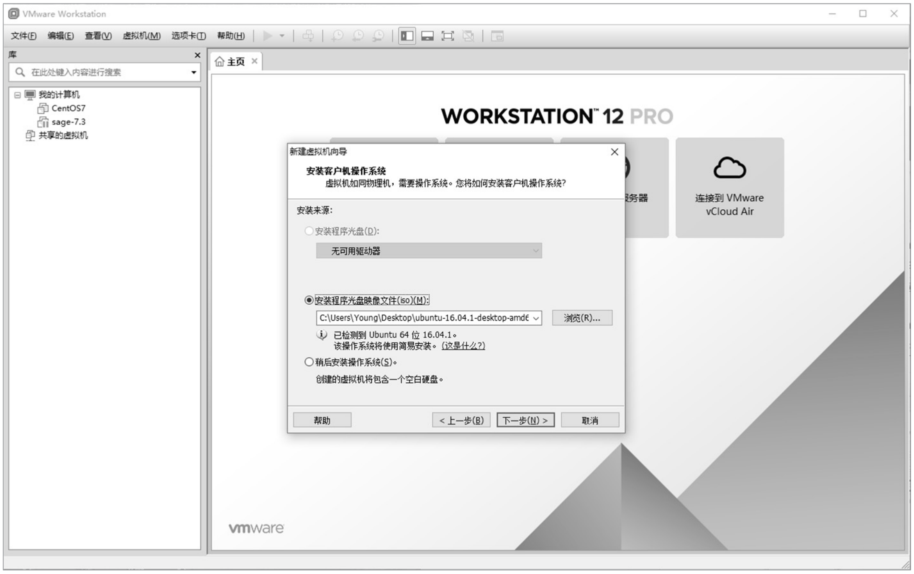


图 A-24 安装 Ubuntu（3）

这个版本的 VMware Workstation 支持简易安装模式。为了讲解方便，在这里把 Linux 虚拟机的全名、用户名、密码都指定为“data”，如图 A–25 所示。

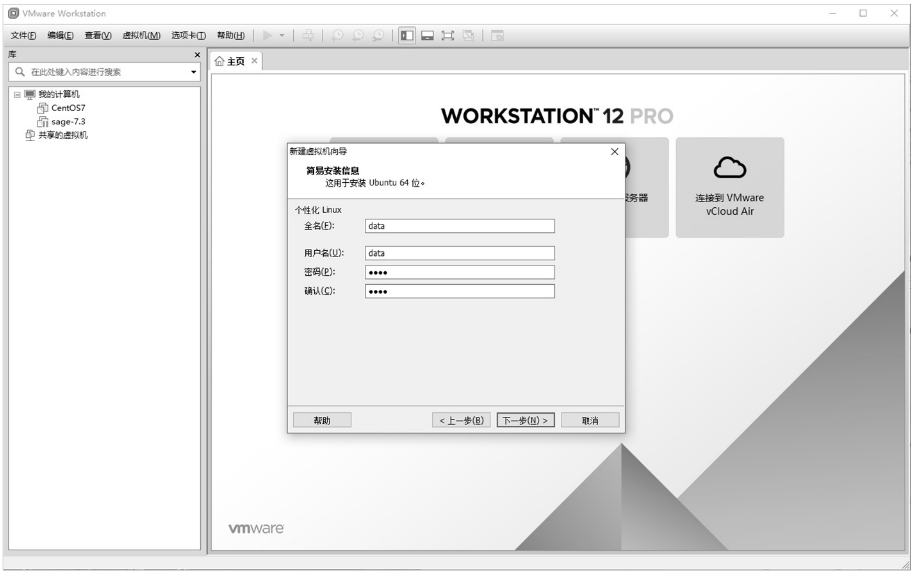


图 A–25　安装 Ubuntu（4）

指定虚拟机的名称（这是在 Ubuntu 管理中使用的名称，不是其主机名）和安装位置，如图 A–26 所示。

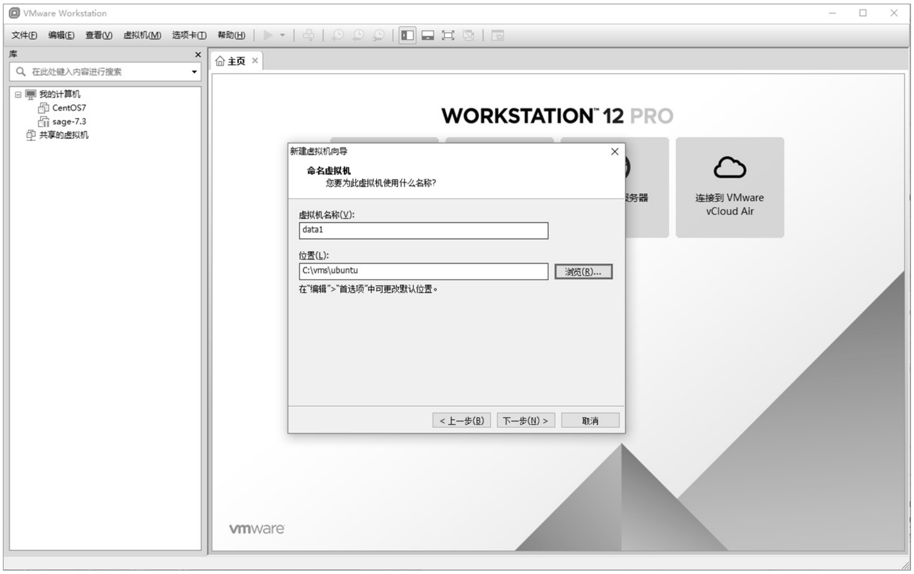


图 A–26　安装 Ubuntu（5）

给虚拟机指定磁盘大小。这里设置的是 100GB，如图 A-27 所示。虚拟机会用按需分配的方式进行动态扩展，所以不用在一开始就因为自己的磁盘空间不够 100GB 而担心程序报错。

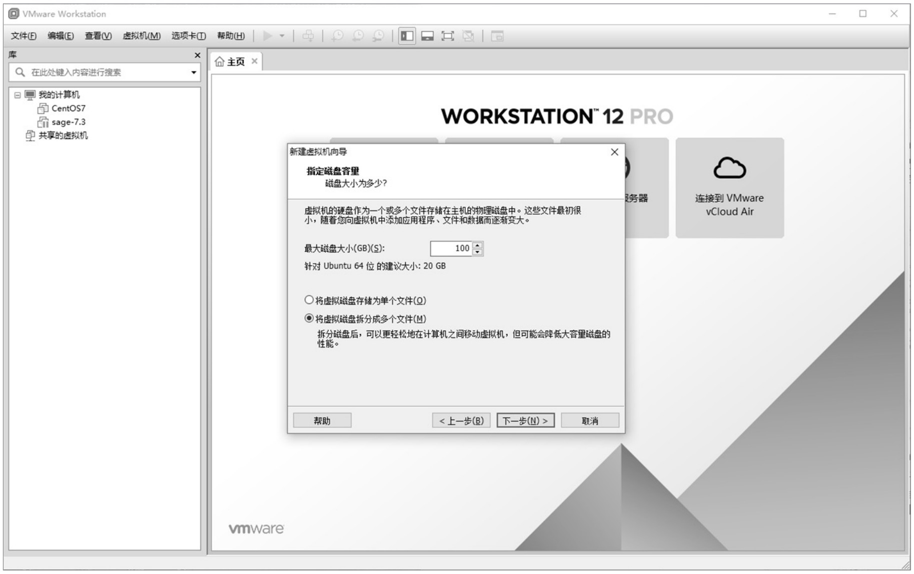


图 A-27 安装 Ubuntu（6）

单击“完成”按钮，引导虚拟机启动，如图 A-28 所示。

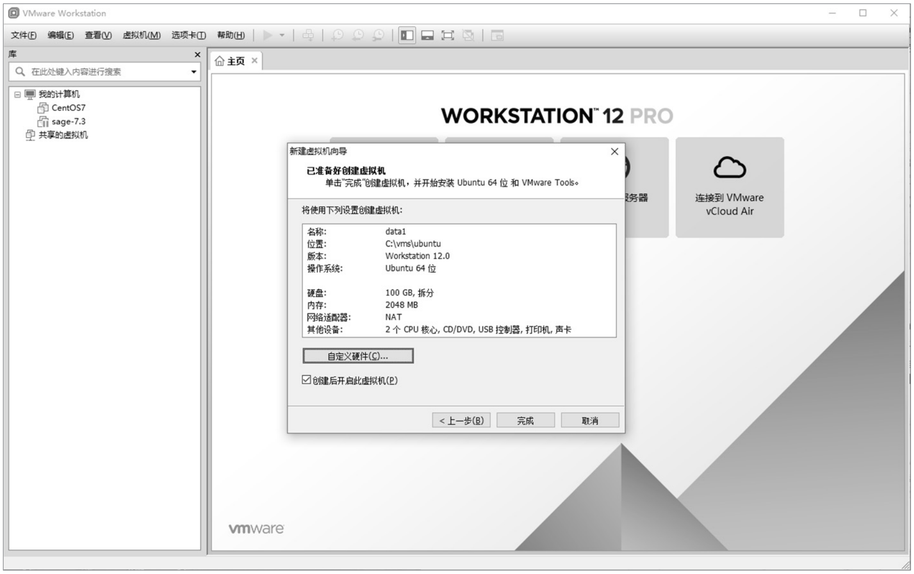


图 A-28 安装 Ubuntu（7）

为了加快安装速度和实验速度，建议在硬件资源充足的情况下赋予虚拟机更多的硬件资源，在这里设置了 4 个 CPU 内核和 8GB 内存。通常情况下，不要把超过 50% 的内存和 CPU 资源划分给虚拟机。

在简易安装模式下，Linux 会自动进行引导，如图 A–29 所示。

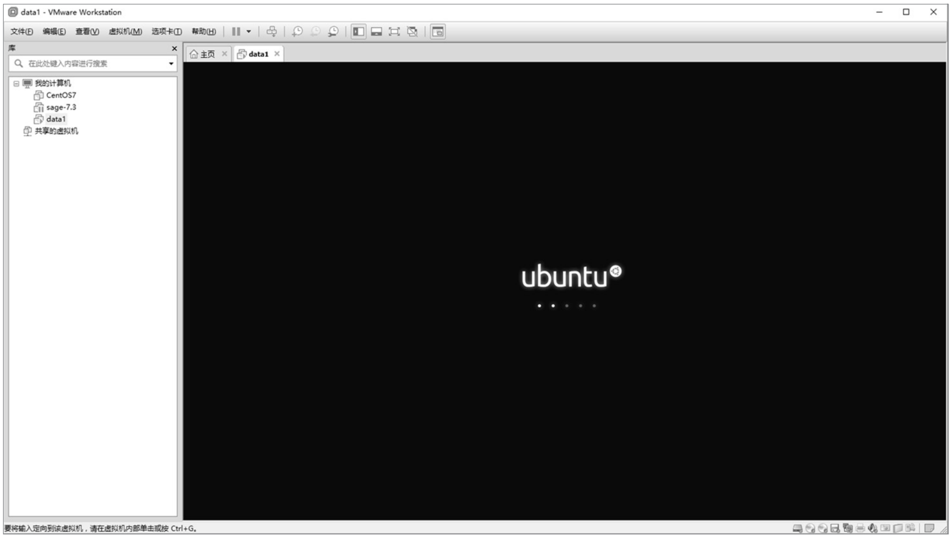


图 A–29　安装 Ubuntu（8）

之后就等待 Linux 自动一步一步安装，不需要人为干预，如图 A–30 所示。

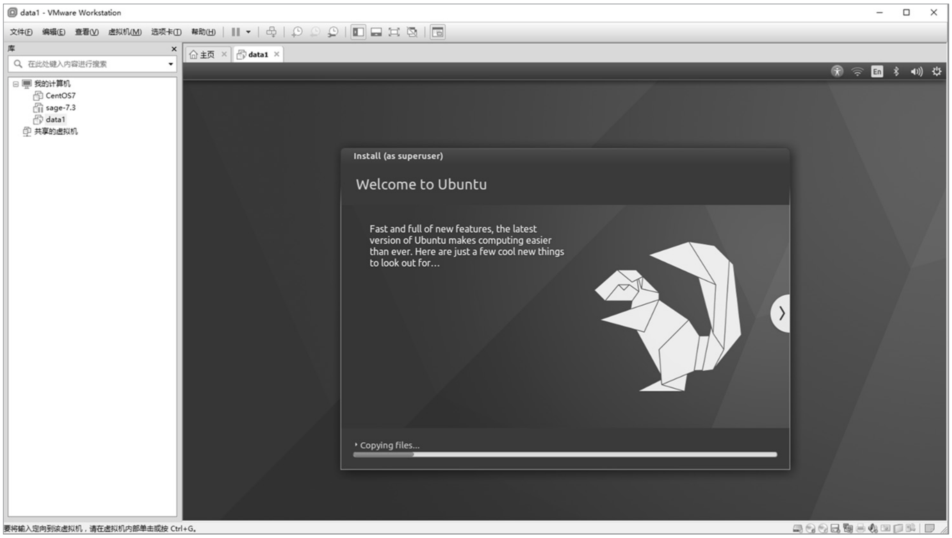


图 A–30　安装 Ubuntu（9）

输入密码“data”，如图 A-31 所示。

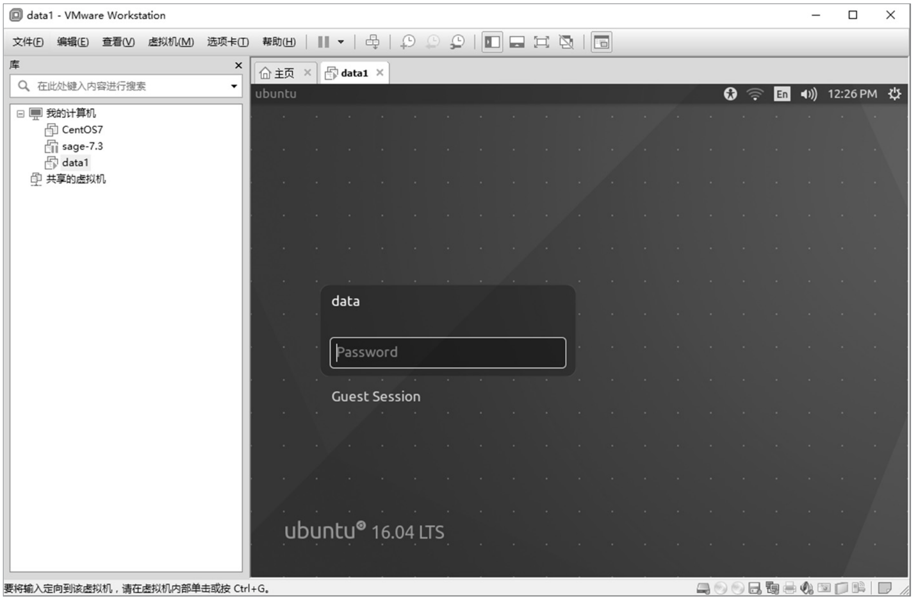


图 A-31 安装 Ubuntu（10）

到此为止，已经进入 Ubuntu 桌面了，如图 A-32 所示。

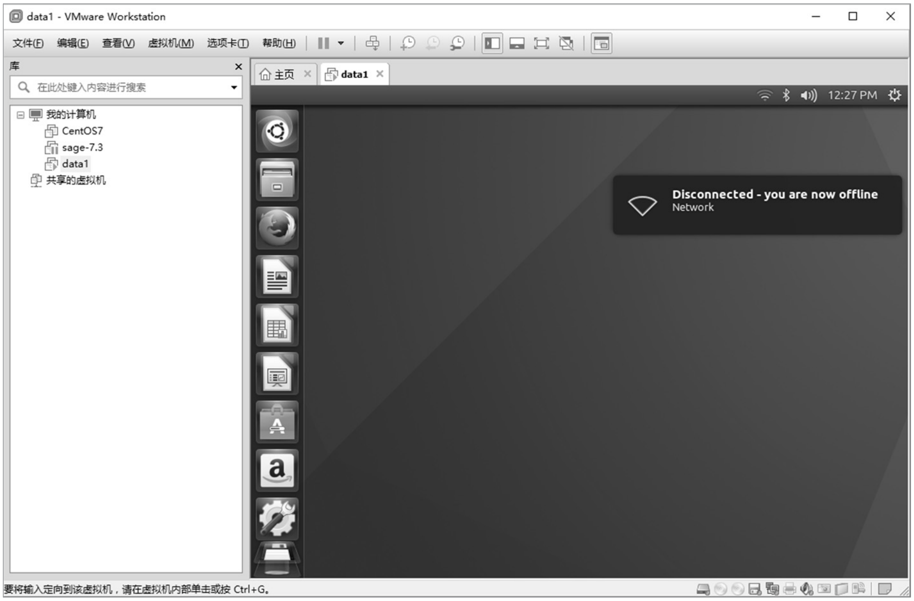


图 A-32 安装 Ubuntu（11）

A.4　Python 语言简介

Python 是一种面向对象的解释型计算机程序设计语言，由 Guido van Rossum 于 1989 年发明，第一个公开版本发行于 1991 年。

Python 是纯粹的自由软件，它的源代码和解释器 CPython 遵循 GPL 协议（GNU General Public License）。

Python 和 C 语言不同，Python 是一种脚本语言。C 语言在写完源代码后需要编译成二进制代码才能执行；Python 则不用，它在生产环境中出现时仍旧是源代码的 .py 文件形式，在执行的瞬间才由 Python 解释器将源代码转换为字节码，然后由 Python 解释器来执行这些字节码。

这种形式的好处是不用考虑平台系统的问题，可以和 Java 语言一样"一次编写，到处执行"，但缺点也是显而易见的，就是每次进行字节码转换和字节码执行的效率不如直接执行二进制高，好在对执行效率苛刻的场合较少。另外，随着计算机硬件能力的提升，执行效率的矛盾也变得不明显了。

和其他计算机语言一样，Python 也有自己的一套语法基础，有顺序、分支、循环、调用的程序组织结构，以及数字、字符串、列表、元组、集合等多种数据类型。

A.4.1　安装 Python

安装 Python 的方法不止一种，在这里只介绍使用 yum 安装 Python 的方式。

按照默认方式安装 CentOS 7 以后，Python 就已经被正确安装了，可以执行如下命令查看。

```
[root@localhost Desktop]# python -V
Python 2.7.5
```

A.4.2　Hello Python

Python 的"Hello World"程序与其他计算机语言没有区别，而且更加简洁，可以直接在交互式编程环境中编写。

```
print ("Hello, Python!");
```

A.4.3　行与缩进

Python 脚本文件和普通的文本文件没有太大区别，一般习惯写成以 .py 作为后缀的方式。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
# 文件名：test.py
```

```
if True:
    print "True"
else:
    print "False"
```

"#"为注释标记，如果在一行中使用了"#"，那么"#"后的内容是不会被解释执行的。下面的"if"和"else"是分支型语句。当"if"后面的内容为"True"（真实）时，将执行"if"所辖部分；否则，将执行"else"所辖部分。

在 Python 语言中，不使用"begin"、"end"或"{}"来表示执行段落的起止。这里的"if"和"else"需要左对齐，并用缩进来表示段落所辖范围。

A.4.4 变量类型

Python 语言中标准的数据类型有 Numbers（数字）、String（字符串）、List（列表）、Tuple（元组）和 Dictionary（字典）。

整数型数字、浮点型数字及字符串类型的赋值和打印操作如下。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

counter = 100 # 赋值整型变量
miles = 1000.0 # 浮点型
name = "John" # 字符串

print counter
print miles
print name
```

列表类型的操作如下。列表很像 Java 语言中的数组，只不过列表允许将不同类型的数据放在同一个列表中，而数组不可以（要求所有的元素类型一致）。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print list # 输出完整列表
print list[0] # 输出列表的第 1 个元素
print list[1:3] # 输出第 2 个至第 3 个元素
print list[2:] # 输出从第 3 个开始至列表末尾的所有元素
print tinylist * 2 # 输出列表 2 次
print list + tinylist # 打印组合的列表
```

元组类型的操作如下。元组类型操作方法和列表很像，但是 Python 语法不允许对元组中的元素进行二次赋值，它相当于只读类型的列表。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
tinytuple = (123, 'john')

print tuple # 输出完整元组
print tuple[0] # 输出元组的第 1 个元素
print tuple[1:3] # 输出第 2 个至第 3 个元素
print tuple[2:] # 输出从第 3 个开始至列表末尾的所有元素
print tinytuple * 2 # 输出元组 2 次
print tuple + tinytuple # 打印组合的元组
```

字典类型的操作如下。字典类型有些像 Java 中的 HashMap，通过主键（key）来访问对应的 value 值，而不是靠下标来访问。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

print dict['one'] # 输出键为'one'的值
print dict[2] # 输出键为 2 的值
print tinydict # 输出完整的字典
print tinydict.keys() # 输出所有键
print tinydict.values() # 输出所有值
```

A.4.5 循环语句

while 循环如下。while 循环后面的条件表示在满足条件的时候执行 while 所辖的程序段。

```
#!/usr/bin/python

count = 0
while (count < 9):
   print 'The count is:', count
   count = count + 1
```

```
print "Good bye!"
```

以上程序表示在 count 小于 9 的情况下执行后面的 2 行语句，但不包括

```
print "Good bye!"
```

这一行。

下面这段程序略显烦琐，不过内容很简单。这是循环的另一种写法——for 循环。for 循环也是一种循环，后面是一个循环范围。这里是一个二重循环，也就是两个循环发生了嵌套——在一个循环的执行中有另一个循环。外层循环是让 num 在 10 到 20 之间循环，内层循环是让 i 在 2 到 num 之间循环。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

for num in range(10,20):  # 迭代 10 到 20 之间的数字
   for i in range(2,num): # 根据因子迭代
      if num%i == 0:      # 确定第 1 个因子
         j=num/i          # 计算第 2 个因子
         print '%d 等于 %d * %d' % (num,i,j)
         break            # 跳出当前循环
   else:                  # 循环的 else 部分
      print num, '是一个质数'
```

A.4.6 函数

函数是最小的代码段封装单位，关键字是“def”。“def”后面的“printme”是函数名，“str”是参数名称。下面这个函数的内容是直接打印传入的变量值，最后两句是对函数的调用。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# 定义函数
def printme( str ):
   "打印任何传入的字符串"
   print str;
   return;

# 调用函数
printme("我要调用用户自定义函数!");
printme("再次调用同一函数");
```

A.4.7　模块

模块是一种大单位的代码段集合。例如，一个 support.py 文件中有多个函数定义，其中一个叫作 print_func 函数。在不对 support.py 这个模块进行引用时，不能调用 print_func 函数。

在下面这段代码中，“import support”表示导入 support.py 模块，“support.print_func("Zara")”表示调用 support 中的 print 函数并传入变量 "Zara" 作为参数。

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# 导入模块
import support

# 现在可以调用模块里包含的函数了
support.print_func("Zara")
```

以下是导入模块的另一种写法，区别是它能够导入一个模块的一部分而非全部模块代码。这里是从 fib 模块中只导入 Fibonacci 函数。

```
from fib import Fibonacci
```

A.4.8　小结

以上介绍了 Python 语言的基本语法。如果读者对强大的 Python 所支持的其他内容感兴趣，可以找一些专门介绍 Python 的资料来学习。

在本书所列举的示例代码中，涉及的库如下。

（1）NumPy

NumPy 系统是 Python 的一种开源的数值计算扩展库。它提供了许多高级的数值编程工具，例如矩阵数据类型、矢量处理及精密的运算库。NumPy 为进行严格的数字处理而生，多为大型金融公司及核心科学计算组织使用，例如 Lawrence Livermore、NASA（用它处理一些本来使用 C++、Fortran 或 Matlab 等处理的任务）。

（2）matplotlib

matplotlib 是一个专业的绘图工具库，官方网址为 http://matplotlib.org/。它调用简单，使用非常方便，是配合 Python 进行数据挖掘和报表制作的利器。

（3）SciPy

SciPy 是一款专为科学和工程设计的方便易用的 Python 工具包。它提供的内容很丰富，包括文件输入/输出、特殊函数、线性代数运算、快速傅里叶变换、统计与随机、微分和积分、图像处理等诸多封装内容。

SciPy 的官方网址为 http://www.scipy.org/。

（4）Scikit-learn

Scikit-learn 是最著名的 Python 机器学习库之一，在 A.5 节会对其进行详细的介绍。

A.5 Scikit-learn 库简介

如果在进行数据分析的过程中需要用到机器学习算法的话，Scikit-learn 库是一个不错的选择。

Python 的开源社区非常活跃，也有很多和 Java 等开源语言一样的框架和库体系，其中 Scikit-learn（后文简写成“sklearn”）是最著名的 Python 机器学习库之一，其官方网址为 http://scikit-learn.org/stable/。

sklearn 基于 BSD 开源许可证，由 David Cournapeau 于 2007 年发起，目前由社区志愿者维护。经过多年的积累，整个项目的内容已经相当丰富，目前最新的稳定版是 0.17。sklearn 用户手册的内容也很友好，覆盖全面，包括有监督的学习（分类）、无监督的学习（聚类）、模型选择与评价、数据集转换、数据集提取应用（数据示例下载）、大规模计算策略和计算效率 7 个部分。

本书中的机器学习算法大多使用 sklearn 库实现，主要涉及 Supervised learning 和 Unsupervised learning 两部分，相信它能帮助我们在实际应用中提高生产效率。

在使用 sklearn 库之前，请确认 Python 已经安装。安装 Python 的方法不止一种，在这里只介绍使用 yum 安装 Python 的方式。

在 A.4.1 节介绍过，按照默认方式安装 CentOS 7 以后，Python 已经被正确安装（如果能正确提示 Python 的版本，则表示 Python 已被正确安装）。如果发现 Python 未被正确安装，可以使用 CentOS 自带的包管理工具 yum 进行安装，方法如下。

```
[root@localhost Desktop]# yum install -y python
```

之后就可以安装 sklearn 了。方法也很简单，只要连接互联网，直接使用 pip 安装命令即可。

```
$ pip install -U scikit-learn
```

A.6 安装 Theano

Theano 是一款基于 Python 的深度学习框架，性能良好，目前的最新版本是 0.8.2，官方网址为 http://www.deeplearning.net/software/theano/。

Theano 支持目前主流的神经网络结构，包括全连接前馈网络、卷积网络、循环神经网络等。

要安装 Theano，首先要安装其基本包依赖。如果是 Python 2，则需要安装 Python 2.6 及以上

版本。如果是 Python 3，则需要安装 Python 3.3 及以上版本。此外，还需要安装 NumPy 1.7.1、SciPy 0.11 及以上版本。

使用 root 用户权限下载 Theano。

```
wget https://github.com/Theano/Theano/archive/master.zip
```

解压缩，命令如下。

```
unzip master.zip
```

安装 Theano。

```
cd Theano-master
pip install Theano –user
```

A.7 安装 Keras

Keras 是一个用 Python 实现的可以应用于 TensorFlow 或 Theano 的神经网络库，有良好的人机接口，使用方便，其官方网址为 https://keras.io/。

Keras 需要 Python 2.7 ~ 3.5 版本的支持。在安装之前，需要安装 TensorFlow 或 Theano。

使用 root 用户权限下载 Keras。

```
wget https://github.com/fchollet/keras/archive/master.zip
```

解压缩，命令如下。

```
unzip master.zip
```

安装 Keras。

```
cd keras-master
python setup.py install
pip install keras
```

A.8 安装 MySQL

在 CentOS 中打开一个 Terminal，输入如下代码。

```
Wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm
```

安装 MySQL。

```
rpm -ivh mysql-community-release-el7-5.noarch.rpm
```

在这之后，“/etc/yum.repo.d/”目录下会多出 2 个文件，分别是 mysql–community.repo 和 mysql–community–source.repo。

使用 yum 安装 MySQL。

```
yum install mysql-server
```

启动 MySQL 服务器程序。

```
service mysqld start
```

使用客户端对服务器进行连接。

```
mysql -u root
```

更改 root 用户的登录密码。

```
[root@localhost ~]# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.33 MySQL Community Server (GPL)
mysql> use mysql;
Database changed

mysql> update user set password=password('data') where user='root';
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> exit;
Bye
```

再次登录 MySQL 服务器。

```
mysql -u root -p
```

输入密码“data”，此时屏幕上会显示如下内容。

```
[root@localhost ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
```

```
Server version: 5.6.33 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

现在，MySQL 服务器已经可以用 root 用户和密码“data”进行连接了。

A.9 安装 MySQL-Python 驱动

访问 https://pypi.python.org/pypi/MySQL-python/1.2.5#downloads，下载 MySQL-python-1.2.5.zip 文件，并将其放到虚拟机的桌面上。

解压缩 MySQL-python-1.2.5.zip。

```
unzip /root/Desktop/MySQL-python-1.2.5.zip
```

进入解压缩后的路径。

```
cd /root/MySQL-python-1.2.5
```

安装 mysql-devel 包。

```
yum install mysql-devel
```

安装 python-devel 包。

```
yum install python-devel
```

安装 MySQLdb。

```
python setup.py build
python setup.py install
```

A.10 MT4 平台简介

MetaTrader 4（MT4）是 MetaQuotes 公司专为金融中介机构定制的网上交易平台，旨在帮助金融中介机构为其客户提供外汇、差价合约、股票和期货市场等的金融产品交易平台。通过 MT4，用户可以开展自己的金融中介业务，为全球金融市场提供交易服务，通过互联网为经纪公司提供全方位的服务（包括后台支持等）。截至 2013 年年底，超过 100 家外汇经纪公司和 30 个国家的银行已经选择 MT4 作为网络交易平台。

MT4 客户端为自行交易者提供了实时模式的交易功能和技术指标。交易平台为交易者提供了强大的功能，包括各种交易执行功能、图表、大量技术指标和曲线、自定义指标及脚本。

在百度搜索引擎中输入“MT4”，会得到很多支持 MT4 的做市商信息，如图 A-33 所示。

图 A-33 搜索结果

任意选取其中的一个完成注册。然后，注册时填写的邮箱会收到一封邮件，里面包括用户名、密码、服务器地址及下载地址（http://www.forexchinese.com/Downloads/MT/forexmt4.exe），如图 A-34 和图 A-35 所示。

由于本书不介绍在线交易的相关内容，所以只需要下载 MT4 平台并导入数据即可。

图 A–34　下载 MT4

图 A–35　开通 MT4

双击安装刚刚下载的 MT4 平台安装文件，勾选“是的，我同意该许可证协议的全部条款。”复选框，然后单击“下一步”按钮，如图 A-36 所示。

等待其自动下载所需的文件并完成安装，如图 A-37 所示。

图 A-36　同意协议条款

图 A-37　等待安装完成

安装完成后，可以连接到 Demo 服务器进行模拟交易，也可以取消连接，如图 A-38 所示。

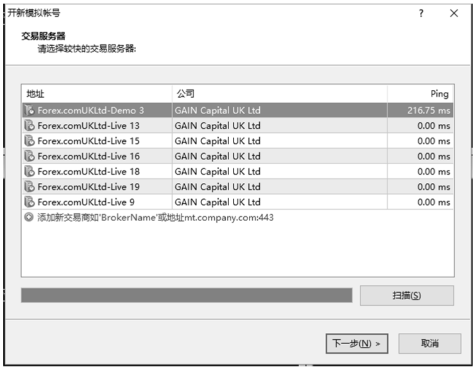


图 A-38　取消连接到服务器

参考文献

[1] 阿基米德. 阿基米德全集（修订版）. 朱恩宽，常心怡，译. 陕西科学技术出版社.

[2] 艾萨克・牛顿. 自然哲学的数学原理. 曾琼瑶，王莹，王美霞，译. 江苏人民出版社.

[3] 同济大学数学系. 高等数学（第六版）上册. 高等教育出版社.

[4] 沃尔特・艾萨克森. 爱因斯坦传. 张卜天，译. 湖南科学技术出版社.

[5] 高扬，卫峥，尹会生. 白话大数据与机器学习. 机械工业出版社.

[6] 吴军. 数学之美. 人民邮电出版社.

[7] 张苍. 九章算术. 曾海龙，译注. 江苏人民出版社.

[8] 翁春生，王浩. 计算内弹道学. 国防工业出版社.

[9] 柏拉图. 理想国. 吴献书，译. 上海三联书店.

[10] Bertrand Russell， Taylor & Francis Group. Principles of Mathematics.

[11] 唐朔飞. 计算机组成原理（第 2 版）. 高等教育出版社.

[12] 李耳. 道德经. 黄彬，温彦国，主编. 中国文史出版社.

[13] 墨翟. 墨子. 李小龙，译. 中华书局出版社.

[14] 班固. 汉书. 中华书局出版社.

[15] 范晔. 后汉书. 线装书局出版社.

[16] 袁枚. 古典名著聚珍文库：随园诗话. 浙江古籍出版社.

[17] 杨建邺. 杨振宁传. 生活・读书・新知三联书店.

[18] 威廉・配第. 政治算数. 商务印书馆.

[19] 贾俊平，何晓群，金勇进. 统计学. 中国人民大学出版社.

[20] Simon Haykin. 神经网络与机器学习. 申富饶，译. 机械工业出版社.

[21] Thomas H.Cormen，Charles E.Leiserson，Ronald L.Rivest，Clifford Stein. 算法导论. 殷建平，徐云，王刚，等译. 机械工业出版社.

[22] 葛立恒，高德纳，帕塔许尼克. 具体数学. 张明尧，张凡，译. 人民邮电出版社.

[23] Jiawei Han，Micheling Kamber，Jian Pei. 数据挖掘概念与技术. 范明，孟小峰，译. 机械工业出版社.

[24] 严蔚敏，吴伟民. 数据结构. 清华大学出版社.
[25] 科弗，托马斯. 信息论基础. 阮吉寿，张华，译. 机械工业出版社.
[26] 周炯槃. 通信原理（第 3 版）. 北京邮电大学出版社.
[27] 张天荣. 蝴蝶效应之谜：走近分形与混沌. 清华大学出版社.
[28] 齐亚乌丁・萨达尔. 混沌学. 梅静，译. 当代中国出版社.
[29] Michal Fekan. Bifurcation and Chaos in Discontinuous and Continuous Systems. 高等教育出版社.
[30] 唐纳德・拉姆斯菲尔德. 已知与未知：美前国防部长拉姆斯菲尔德回忆录. 魏骍，译. 华文出版社.
[31] Frank R. Giordano，William P.Fox，Steven B.Horton. 数学建模（原书第 5 版）. 叶其孝，姜启源，等译. 机械工业出版社.
[32] Bruce Schneier. 应用密码学：协议、算法与 C 源程序. 吴世忠，祝世雄，张文政，等译. 机械工业出版社.
[33] 李航. 统计学习方法. 清华大学出版社.
[34] Ian H.Witten，Alistair Moffat，Timothy C.Bell. 管理海量数据：压缩、索引和查询. 梁斌，杨青，译. 电子工业出版社.
[35] 张晓明. Oracle DBA 突击：帮你赢得一份 DBA 职位. 人民邮电出版社.
[36] 博韦，西斯特. 深入理解 LINUX 内核（第 3 版）. 陈莉君，张琼声，张宏伟，译. 中国电力出版社.
[37] W.Richard Stevens. TCP/IP 详解卷 1：协议. 范建华，胥光辉，张涛，等译. 机械工业出版社.
[38] 库罗斯，罗斯. 计算机网络：自顶向下方法（原书第 6 版）. 机械工业出版社.
[39] Randal E. Bryant，David R.O Hallaren. 计算机科学丛书：深入理解计算机系统（原书第 2 版）. 机械工业出版社.
[40] Tom White. Hadoop 权威指南（第 3 版 修订版）. 华东师范大学数据科学与工程学院，译. 清华大学出版社.
[41] James Baldwin. Fifty Famous People. Yesterday’s Classics.
[42] 周淑贞. 气象学与气候学（第 3 版）. 高等教育出版社.
[43] Walter Rudin. 泛函分析（原书第 2 版）. 机械工业出版社.
[44] Douglas M. Bates，Donald G. Watts. 非线性回归分析及其应用. 韦博成，万方焕，朱宏图，译. 中国统计出版社.
[45] 尼尔・J・萨尔金德. 爱上统计学（第 2 版）. 史玲玲，译. 重庆大学出版社.
[46] 韩非. 韩非子. 高华平，王齐洲，张三夕，译著. 中华书局.

[47] 斯蒂芬・P・罗宾斯，玛丽・库尔特. 管理学（第 11 版本）. 李原，等译. 中国人民大学出版社.

[48] 唐浩明. 曾国藩. 广东人民出版社.

[49] 宋鸿兵. 货币战争（升级版）. 中信出版社.

[50] 亚当・斯密. 国富论. 郭大力，王亚南，译. 译林出版社.

[51] Steven W. Smith, Ph.D. The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Pub, 1997.

[52] Jacob Ziv, Abraham Lempel. A Universal Algorithm for Sequential Data Compression. 1977.

[53] Johs Schmidt. The Breeding Places of the Eel. 1921.

[54] Claude Elwood Shannon. A Mathematical Theory of Communication. The Bell System Technical Journal. 1948.

[55] George Armitage Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. 1956.

[56] International Technical Support Organization. DB2 – Quick Upper Bound Estimate An Application Design Methodology. 1995.

[57] C.Radhakrishna Rao. Statistics and Truth. 1997.

[58] Nageswara Rao Thota, Srinivasa Kumar Devireddy. Image Compression Using Discrete Cosine Transform. Georgian Electronic Scientific Journal: Computer Science and Telecommunications, 2008. No.3: 17.

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：（010）88254396；（010）88258888

传　　真：（010）88254397

E-mail：　dbqq@phei.com.cn

通信地址：北京市万寿路173信箱　电子工业出版社总编办公室

邮　　编：100036